FAST and NESTOR: How to Exploit Annotation Hierarchies

Nicola Ferro and Gianmaria Silvello

Department of Information Engineering, University of Padua, Italy
{ferro, silvello}@dei.unipd.it

Abstract. In this paper we present the annotation model implemented by *Flexible Annotation Service Tool (FAST)* and the set-theoretical data models defined in the *NEsted SeTs for Object hieRarchies (NESTOR)* framework. We show how annotations assume a tree structure that can be exploited by NESTOR to improve access and exchange of Digital Objects (DOs) between *Digital Librarys (DLs)* in a distributed environment.

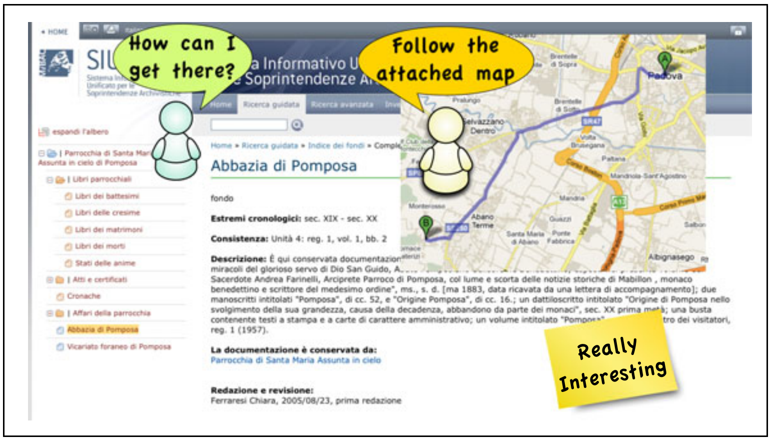
1 Motivations

DLs are getting the preponderant mean to manage, exchange and retrieve cultural heritage resources. DLs can be seen as tools for managing information resources of different kinds of organizations ranging from libraries, and museums to archives. In these different contexts, DLs permit the management of wide and different corpora of resources which range from books and archival documents to multimedia resources as pointed out in [8].

Furthermore, DL are not only systems which permit the users to manage, exchange and retrieve digital objects or metadata, they are increasingly becoming part of the user's work. DL can enable the intellectual production process and support user cooperation and exchange of ideas. In this way, DL not only foster access to knowledge, but they are also part of knowledge creation and evolution. The evolution and transmission of knowledge has always been an interactive process between scientists or field experts and annotations have always been one of the main tools for this kind of interaction. In the digital era, annotations are still means of intellectual collaboration and in DL they are considered first-class digital objects [7]. They are also adopted in a variety of different contexts, such as content enrichment, data curation, collaborative and learning applications, and social networks, as well as in various information management systems, such as the Web (semantic and not), and databases.

As an example, in Figure 1 we can see different kinds of annotations commenting an archival document managed by an archival system[1]. Documents in an archive are organized in a hierarchy [4] and the annotations may need to be attached both to the content - the actual documents - and to the structure - the

[1] Sistema Informativo Unificato per le Soprintendenze Archivistiche (SIUSA) http://siusa.archivi.beniculturali.it/



| Parrocchia di Santa Mari Assunta in cielo di Pomposa
| Libri parrocchiali
Libri dei battesimi
Libri delle cresime
Libri dei matrimoni
Libri dei morti
Stati delle anime
| Atti e certificati
Cronache
| Affari della parrocchia
Abbazia di Pomposa
Vicariato foraneo di Pomposa

Home » Ricerca guidata » Indice dei fondi » Comple

Abbazia di Pomposa

fondo

Estremi cronologici: sec. XIX - sec. XX

Consistenza: Unità 4: reg. 1, vol. 1, bb. 2

Descrizione: È qui conservata documentazion
miracoli del glorioso servo di Dio San Guido, A.
Sacerdote Andrea Farinelli, Arciprete Parroco di Pomposa, col lume e scorta delle notizie storiche di Mabillon , monaco benedettino e scrittore del medesimo ordine", ms., s. d. [ma 1883, data ricavata da una lettera di accompagnamento]; due manoscritti intitolati "Pomposa", di cc. 52, e "Origine Pomposa", di cc. 16.; un dattiloscritto intitolato "Origine di Pomposa nello svolgimento della sua grandezza, causa della decadenza, abbandono da parte dei monaci", sec. XX prima metà; una busta contenente testi a stampa e a carte di carattere amministrativo; un volume intitolato "Pompos tro dei visitatori, reg. 1 (1957).

La documentazione è conservata da:
Parrocchia di Santa Maria Assunta in cielo

Redazione e revisione:
Ferraresi Chiara, 2005/08/23, prima redazione

Fig. 1. Different kinds of annotation annotating an archival document

relationships between the documents - of an archive. Furthermore, in Figure 1 we can see that an annotation can be a textual comment (i.e. "really interesting" which annotates the content of the document) or the expression of a user information need (i.e. "How can I get there?" which annotates an element of the archival structure); the map is an annotation provided by a user and conceptually attached to another annotation and not to an archival component.

The previous example shows that annotations are a quite complex concept comprising a number of different aspects and in order to deal with this heterogeneity a formal model for digital annotations has been proposed in [1] and this model has been adopted by the FAST annotation service for representing and managing annotations. Furthermore, we pointed out the importance of annotating both the content and the structure of digital objects; in [6] we presented the NESTOR Framework which is based on two set data models alternative to the tree that permit us to model hierarchies handling structure and content in an independent way. The clear distinction and autonomous treatment of content and structure elements in the NESTOR Framework ease the adoption of services which consider the structural elements defining the organization of the objects of interest as relevant as the objects themselves. Indeed, it can be exploited by FAST enabling a natural way to annotate both the content and the structure of hierarchies of objects. Moreover, we shall see that also annotations can be shaped into a hierarchy and thus modeled throughout the NESTOR Framework as well enabling a uniform representation of both annotated objects and annotations.

The use of NESTOR with annotations concerns: the data structure used to attach annotations to hierarchies and contents, the way in which annotations are accessed and exchanged in a distributed environment and the annotation search strategies.

The paper is organized as follows: Section 2 points out the characteristics of FAST annotation model highlighting the hypertext created by annotations in a DL and their hierarchical structure. Section 3 introduces the theoretical foundations of the NESTOR framework. Section 4 describes how NESTOR can be applied to the presented annotation model and points out the advantages of this approach. Finally, Section 5 draws some final remarks.

2 FAST Annotation Model

FAST adopts and implements the formal model for annotations proposed by [1] which has been also embedded in the reference model for digital libraries developed by DELOS, the European network of excellence on digital libraries [2]. According to this model an annotation is a compound multimedia object constituted by different signs of annotation which materialize the annotation itself; for example, we can have textual signs, which contain the textual content of the annotation, image signs, if the annotation is made up of images, and so on. In turn, each sign is characterized by one or more meanings of annotation which specify the semantics of the sign; for example, we can have a sign whose meaning corresponds to the title field in the Dublin Core (DC) metadata schema[2], in the case of a metadatum annotation, or we can a sign carrying a question of the author about a document whose meaning may be question or similar. Every annotation is uniquely identified by the pair (namespace, identifier).

In the following, we need a terminology to distinguish between two kinds of digital objects: the generic ones managed by a digital library or available in the Web, which we call *documents*, and the ones that are *annotations*. Therefore, when we use the generic term *digital object*, we mean a digital object that can be either a document or an annotation.

Annotations can be associated to a digital object, that can be both a document or an annotation, by two types of link:

– **annotate link:** it permits to link an annotation to a part of a digital object. By means of annotate link an annotation can annotate one or more parts of a digital object expressing *intra-digital object* relationships between the different parts of the annotated digital object. An important constraint is that an annotation can annotate one and only one digital object.
– **relate-to link:** it is intended to allow an annotation only to relate to one or more parts of other digital objects, but not the annotated one. Therefore, this kind of link lets the annotation express *inter-digital object* relationships, meaning that the annotation creates a relationship between the annotated digital object and the other digital objects related to it. By means of relate-to links an annotation can link more digital objects.

From these definitions annotations can be seen as linking means between digital objects. Annotations permit us to create new relationships between the components of a digital objects, between different digital objects of the same DL or between digital objects belonging to different DLs. As shown in [1] the set of digital objects and annotations forms a labeled directed acyclic graph called document-annotation hypertext.

[2] http://www.dublincore.org/

Definition 1. *Let A be the set of annotations, D the set of documents, and $DO = A \cup D$ the set of digital objects, which are either annotations or documents. The* ***document-annotation hypertext*** *is a labeled directed graph*

$$H_{da} = (DO, E_{da} \subseteq A \times DO, l_{da}) \tag{2.1}$$

where:

– *$DO = A \cup D$ is the set of vertices;*
– *E_{da} is the set of edges;*
– *$l_{da} : E_{da} \to LT$, with $LT = \{annotate, relate\text{-}to\}$, is the labeling function which associates the corresponding link type to each edge.*

It has been proved in [1] that H_{da} does not contain loops and cycles. Furthermore, we know that each annotation must annotate only one digital object, thus for each document there is a **unique tree of annotations** constituted by "annotate" edges that can be rooted in the document. For each annotation we can determine the unique path to the document root of the tree at which the annotation belongs. In this work we aim to point out the backbone of annotations which is composed by the "annotate" links; we are interested in independent threads of annotations rather than in the general structure of the annotation graph. The following proposition which is extensively described and proved in [1] outlines this aspect of annotations.

Proposition 1. *Let $H^{'}_{da} = (DO^{'}, E^{'}_{da})$ be the subgraph of H_{da}, such that:*

– *$E^{'}_{da} = \{e \in E_{da} \mid l_{da}(e) = Annotate\}$;*
– *$DO^{'} = \{do \in DO \mid \exists e^{'} \in E^{'}_{da}, e^{'} = (a, do)\}$*

$H^{'}_{da}$ is the subgraph whose edges are of the kind, Annotate, and whose vertices are incident with at least one of these edges. Let $H^{''}_{da} = (DO^{''}, E^{''}_{da})$ be the underlying graph of $H^{'}_{da}$, which is the undirected version of $H^{'}_{da}$. The following properties hold:

– *$H^{''}_{da}$ is a forest;*
– *every tree in $H^{''}_{da}$ contains a unique document vertex d.*

In Figure 2 we can see an example of document-annotation hypertext, the directed acyclic graphs ($H^{'}_{da}$) formed by the "annotate" links with the "relate-to" links are removed and the forest $H^{''}_{da}$.

The formal model for annotation provided a sound basis for designing and developing an XML Schema for the FAST annotation service. The FAST XSchema[3] allows annotations and related entities to be represented and exchanged into a well-defined XML format. The FAST XSchema encodes both the content and the metadata about the annotation. For instance, it encodes the data about the user who created the annotation and about the groups sharing the annotation; at the same time the FAST XSchema permits us to encode the "annotate" and the "relate-to" links information.

[3] http://ims.dei.unipd.it/xml/fast-schema-instance

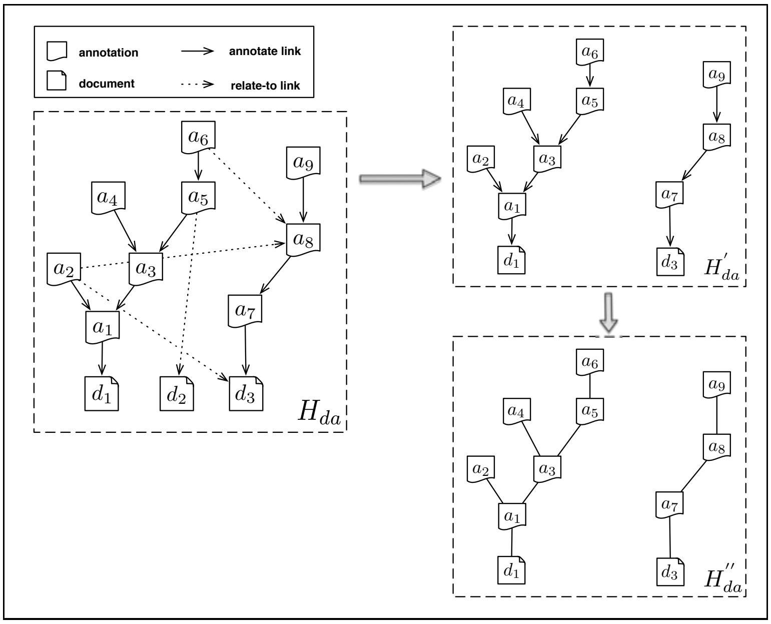


Fig. 2. A document-annotation hypertext H_{da} and the forest H''_{da} composed by two trees created considering only the "annotate" links

3 The NESTOR Framework

We propose two set data models called *Nested Set Model* (NS-M) and *Inverse Nested Set Model* (INS-M) based on an organization of nested sets. The foundational idea behind these set data models is that an opportune set organization can maintain all the features of a tree data structure with the addition of some new relevant functionalities. We define these functionalities in terms of flexibility of the model, rapid selection and isolation of easily specified subsets of data and extraction of only those data necessary to satisfy specific needs.

The most intuitive way to understand how these models work is to relate them to the well-know tree data structure. Thus, we informally present the two data models by means of examples of mapping between them and a sample tree. The first model we present is the **Nested Set Model** (NS-M). An organization of sets in the NS-M is a collection of sets in which any pair of sets is either disjoint or one contains the other. In figure 3 (b) we can see a tree mapped into a NS-M represented by the means of an Eulero-Venn diagram. We can see that each node of the tree is mapped into a set, where child nodes become *proper subsets* of the set created from the parent node. Every set is subset of at least of one set; the set corresponding to the tree root is the only set without any supersets and every set in the hierarchy is subset of the root set. The external nodes are sets with no subsets. The tree structure is maintained thanks to the nested organization and the relationships between the sets are expressed by the set inclusion order.

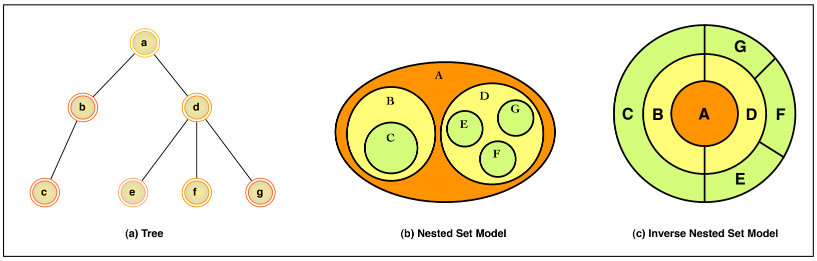


Fig. 3. (a) A tree. (b) The Euler-Venn Diagram of a NS-M. (c) the Doc-Ball representation of a INS-M.

The second data model is the **Inverse Nested Set Model** (INS-M). We can say that a tree is mapped into the INS-M transforming each node into a set, where each parent node becomes a subset of the sets created from its children. The set created from the tree's root is the only set with no subsets and the root set is a proper subset of all the sets in the set organization. The leaves are the sets with no supersets and they are sets containing all the sets created from the nodes composing the tree path from a leaf to the root. An important aspect of INS-M is that the intersection of every couple of sets obtained from two nodes is always a set representing a node in the tree. The intersection of all the sets in the INS-M is the set mapped from the root of the tree.

In Figure 3 (c) we can see a tree mapped into a INS-M represented throughout a DocBall representation [11]. The representation of the INS-M by means of the Euler-Venn diagrams is not very expressive and can be confusing for the reader, for these reasons we use the DocBall representation. We exploit the DocBall ability to show the structure of an object and to represent the "*inclusion order of one or more elements in another one*" [11]. The DocBall is composed of a set of circular sectors arranged in concentric rings as shown in Figure 3 (c). In a DocBall each ring represents a level of the hierarchy with the center (level 0) representing the root. In a ring, the circular sectors represent the nodes in the corresponding level. We use the DocBall to represent the INS-M, thus for us each circular sector corresponds to a set.

It is worthwhile for the rest of the work to define some basic concepts of set theory: the family of subsets and the subfamily of subsets, with reference to [3] for their treatment. However, we assume the reader is confident with the basic concepts of ZFC axiomatic set theory [9], which we cannot extensively treat here for space reasons.

Definition 2. *Let A be a set, I a non-empty set and $\mathcal{C}$ a collection of subsets of A. Then a bijective function $\mathcal{A} : I \longrightarrow \mathcal{C}$ is a* ***family*** *of subsets of A. We call I the* ***index*** *set and we say that the collection $\mathcal{C}$ is* ***indexed*** *by I.*

We use the following notation $\{A_i\}_{i \in I}$ to indicate the family $\mathcal{A}$; the notation $A_i \in \{A_i\}_{i \in I}$ means that $\exists\ i \in I \mid \mathcal{A}(i) = A_i$. We call **subfamily** of $\{A_i\}_{i \in I}$ the **restriction** of $\mathcal{A}$ to $J \subseteq I$ and we denote this with $\{B_j\}_{j \in J} \subseteq \{A_i\}_{i \in I}$.

Definition 3. *Let $\{A_i\}_{i \in I}$ be a family. We define $\{A_i\}_{i \in I}$ to be a* ***linearly ordered family*** *if $\forall A_j, A_k \in \{A_i\}_{i \in I}, A_j \subseteq A_k \vee A_k \subseteq A_j$.*

Furthermore, we can say that a family $\{A_i\}_{i \in I}$ is a linearly ordered family if every two sets in $\{A_i\}_{i \in I}$ are comparable. In literature a linearly ordered family is also called a chain.

Definition 4. *Let $\{A_i\}_{i \in I}$ be a family . We define $\{A_i\}_{i \in I}$ to be a* ***topped family*** *if $\exists A_k \in \{A_i\}_{i \in I} \mid \forall A_j \in \{A_i\}_{i \in I}, A_j \subseteq A_k$. If $\nexists A_k \in \{A_i\}_{i \in I} \mid \forall A_j \in \{A_i\}_{i \in I}, A_j \subseteq A_k$ then $\{A_i\}_{i \in I}$ is defined to be a* ***topless family****.*

Definition 5. *Let A be a set and let $\{A_i\}_{i \in I}$ be a family. Then $\{A_i\}_{i \in I}$ is a* ***Nested Set*** *family if:*

$$A \in \{A_i\}_{i \in I}, \tag{3.1}$$

$$\emptyset \notin \{A_i\}_{i \in I}, \tag{3.2}$$

$$\forall A_h, A_k \in \{A_i\}_{i \in I}, h \neq k \mid A_h \cap A_k \neq \emptyset \Rightarrow A_h \subset A_k \vee A_k \subset A_h. \tag{3.3}$$

Thus, we define a Nested Set family (NS-F) as a family where three conditions must hold. The first condition (3.1) states that set A which contains all the sets in the family must belong to the NS-F. The second condition (3.2) states that the empty-set does not belong to the NS-F and the last condition (3.3) states that the intersection of every couple of distinct sets in the NS-F is not the empty-set only if one set is a proper subset of the other one.

Theorem 2. *Let $T(V, E)$ be a tree and let Φ be a family where $I = V$ and $\forall v_i \in V$, $V_{v_i} = \Gamma_V^+(v_i)$. Then $\{V_{v_i}\}_{v_i \in V}$ is a Nested Set family.*

This theorem defines how a tree is mapped into a NS-F, the proof and an extensive description can be found in [6].

In the same way we can define the Inverse Nested Set Model (INS-M):

Definition 6. *Let A be a set and let $\{A_i\}_{i \in I}$ be a family and let $\{B_j\}_{j \in J} \subseteq \{A_i\}_{i \in I}$ be a sub-family. Then $\{A_i\}_{i \in I}$ is an* ***Inverse Nested Set*** *family if:*

$$\emptyset \notin \{A_i\}_{i \in I}, \tag{3.4}$$

$$\bigcap_{j \in J} B_j \in \{A_i\}_{i \in I}. \tag{3.5}$$

$$\begin{aligned} &\exists B_k \in \{B_j\}_{j \in J} \mid \forall B_h \in \{B_j\}_{j \in J}, B_h \subseteq B_k \\ &\Rightarrow \forall B_h, B_g \in \{B_j\}_{j \in J}, B_h \subseteq B_g \vee B_g \subseteq B_h. \end{aligned} \tag{3.6}$$

Thus, we define an Inverse Nested Set family (INS-F) as a family where two conditions must hold. The first condition (3.4) states that the empty-set does not belong to the INS-F. The second condition (3.5) states that the intersection of every subfamily of the INS-F belongs to the INS-F itself. Condition 3.6 states that every subfamily of a INS-F can be a topped family only if it is linearly ordered; alternatively, we can say that every subfamily of an INS-F must be a topless family or a chain.

Theorem 3. *Let $T(V, E)$ be a tree and let Ψ be a family where $I = V$ and $\forall v_i \in V$, $V_{v_i} = \Gamma_V^-(v_i)$. Then $\{V_{v_i}\}_{v_i \in V}$ is an Inverse Nested Set family.*

Differently from Theorem 2 we report the proof of this theorem because it is slightly different form the one presented in [6].

Proof. By definition of the set of the ancestors of a node, $\forall v_i \in V$, $|V_{v_i}| = |\Gamma_V^-(v_i)| \geq 1$ and so $\emptyset \notin \{V_{v_i}\}_{v_i \in V}$ (condition 3.4).

Let $\{B_{v_j}\}_{v_j \in J}$ be a subfamily of $\{V_{v_i}\}_{v_i \in V}$. We prove condition 3.5 by induction on the cardinality of J. $|J| = 1$ is the base case and it means that every subfamily $\{B_{v_j}\}_{v_j \in J} \subseteq \{V_{v_i}\}_{v_i \in V}$ is composed only by one set B_{v_1} whose intersection is the set itself and belongs to the family $\{V_{v_i}\}_{v_i \in V}$ by definition.

For $|J| = n-1$ we assume that $\exists\ v_{n-1} \in V \mid \bigcap_{v_j \in J} B_{v_j} = B_{v_{n-1}} \in \{V_{v_i}\}_{v_i \in V}$; equivalently we can say that $\exists\ v_{n-1} \in V \mid \bigcap_{v_j \in J} \Gamma_V^-(v_j) = \Gamma_V^-(v_{n-1})$, thus, $\Gamma_V^-(v_{n-1})$ is a set of nodes that is composed of common ancestors of the $n-1$ considered nodes.

For $|J| = n$, we have to show that $\exists\ v_t \in V \mid \forall\ v_n \in J,\ B_{v_{n-1}} \cap B_{v_n} = B_{v_t} \in \{V_{v_i}\}_{v_i \in V}$. This is equivalent to show that $\exists\ v_t \in V \mid \forall\ v_n \in J,\ \Gamma_V^-(v_{n-1}) \cap \Gamma_V^-(v_n) = \Gamma_V^-(v_t)$.

Ab absurdo suppose that $\exists\ v_n \in J \mid \forall\ v_t \in V,\ \Gamma_V^-(v_{n-1}) \cap \Gamma_V^-(v_n) \neq \Gamma_V^-(v_t)$. This would mean that v_n has no ancestors in J and, consequently, in V; at the same time, this would mean that v_n is an ancestor of no node in J and, consequently, in V. But this means that V is the set of nodes of a forest and not of a tree.

Now, we have to prove condition 3.6. Let $\{B_{v_j}\}_{v_j \in J}$ be a subfamily of $\{V_{v_i}\}_{v_i \in V}$. Ab absurdo suppose that $\exists B_{v_k} \in \{B_{v_j}\}_{v_j \in J} \mid \forall B_{v_h} \in \{B_{v_j}\}_{v_j \in J}$, $B_{v_h} \subseteq B_{v_k} \Rightarrow \exists B_{v_h}, B_{v_g} \in \{B_{v_j}\}_{v_j \in J} \mid B_{v_h} \nsubseteq B_{v_g} \wedge B_{v_g} \nsubseteq B_{V_h}$. This means that $\{B_{v_j}\}_{v_j \in J}$ is a topped but not linearly ordered family.

This means that we can find $B_{v_g}, B_{v_h}, B_{v_k} \in \{B_{v_j}\}_{v_j \in J} \mid ((B_{v_h} \cap B_{v_k} \neq \emptyset) \wedge (B_{v_h} \cup B_{v_k} \subset B_{v_g}) \wedge (B_{v_h} \nsubseteq B_{v_k}) \wedge (B_{v_k} \nsubseteq B_{v_h})) \Rightarrow \exists v_h, v_k, v_g \in V \mid ((\Gamma_V^-(v_h) \cap \Gamma_V^-(v_k) \neq \emptyset) \wedge (\Gamma_V^-(v_h) \cup \Gamma_V^-(v_k) \subseteq \Gamma_V^-(v_g)) \wedge (\Gamma_V^-(v_h) \nsubseteq \Gamma_V^-(v_k)) \wedge (\Gamma_V^-(v_k) \subseteq \Gamma_V^-(v_h)))$. This means that there are two paths from the root of T to v_g, one through v_h and a distinct one through v_k, thus $\delta_V^-(v_g) = 2$ and so T is not a tree. $\square$

4 FAST and NESTOR: A Set-Theoretic View of Annotation Hierarchies

From the data model perspective, in the context of DLs we have to take into account both the organization of digital resources and the organization of annotations. In the Definition 1 we treated both annotations and documents as Digital Objects (DO), it is worthwhile for the rest of the work to maintain this notation and, as a generalization, we indicate as documents every resource type managed by a DL which is not an annotation.

The NESTOR framework can be applied to all DO organizations with a tree structure. In the following we show how NESTOR can be applied to a tree where the nodes are documents and where each node can be the root of a sub-tree of annotations. The union of the document tree and the annotation sub-trees forms a DO tree that can be uniformly mapped into one of the set data models formalized in the NESTOR framework.

Let $T(D_T, E_T)$ be a document tree where $D_T = \{d_i\}, 1 \leq i \leq n$ is the set of documents representing the nodes of the tree and $E_T \subset D_T \times D_T$ is the set of edges connecting the nodes. From Proposition 1 we know that for all $d_i \in D_T$ may exist a tree $H^{''}_{da}[d_i]$ of annotations rooted in d_i; $H^{''}_{da}$ is a forest representing the union of all the trees $H^{''}_{da}[d_i]$. From Definition 1 we know that $DO = D \cup A$, thus we can define the tree $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ where $DO_{TH^{''}} = D_T \cup DO^{''}$ and $E_{TH^{''}} = E_T \cup E^{''}_{da}$. $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ is a DO tree because its nodes may be both documents and annotations. In Figure 4 we can see a document tree, two annotation sub-trees and how the union of these trees forms a DO tree. From this Figure we can see that we have to deal we three trees each of those enabling the access to a different granularity level of information: $T(D_T, E_T)$ permits us to access only the documents managed by a DL, $H^{''}_{da}$ permits us to access only the annotations and the annotated documents (the roots of the annotation trees) and $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ permits us to access the whole resource space composed by documents and annotations.

All the DOs belonging to the tree $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ can be encoded in *eXtensible Markup Language (XML)* files; for instance if $T(D_T, E_T)$ represents an archival tree, each node (document) would represent a division of the archive such as fonds, sub-fonds or series that can be encoded in XML files. The annotations, as well, are encoded following the FAST XSchema and thus are treated as XML files. All the relationships between the nodes of $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ are hard coded

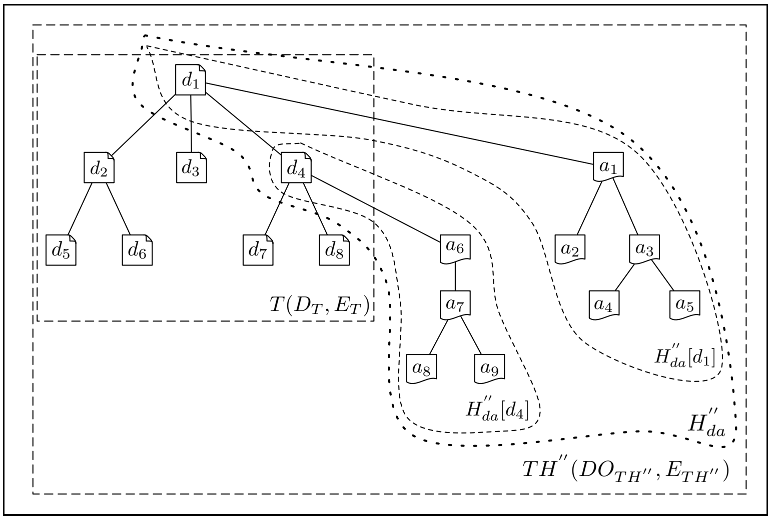


Fig. 4. A sample document tree $T(D_T, E_T)$ with two nodes which are roots of two annotation sub-trees

inside the XML files; indeed, for instance, FAST XSchema permits us to encode the "annotate" links information and a similar approach is adopted by *Encoded Archival Description (EAD)* metadata format [10] in the archival context.

The adoption of the NESTOR framework enables the separation of the information about the hierarchical structure of DOs from their content because the structural links are mapped into inclusion dependencies between sets. In the case of annotations only the "annotate" links can be treated by means of NESTOR because they form a tree structure; instead, the "relate-to" links form a directed acyclic graph that is out of the scope of NESTOR. Following the Theorems 2 and 3 the document tree, the annotation forest and the DO tree can be straightforwardly mapped into an equivalent number of NS-F or INS-F. We define $\{T_i\}_{i \in I}$ to be the family mapped from $T(D_T, E_T)$, $\{H_j^{''}\}_{j \in J}[d_t]$ to be the family mapped from $H_{da}^{''}[d_t]$ where $\mathcal{H}^{''} = \{\{H_j^{''}\}_{j \in J}[d_t]\}$ with $d_t \in D_T$ is the collection of families $\{H_j^{''}\}_{j \in J}$ and $\{TH_k^{''}\}_{k \in K}$to be the family mapped from $TH^{''}(DO_{TH''}, E_{TH''})$. If the mapping from the trees to the families is done following Theorem 2 we obtain a collection of NS-F, as we can see in Figure 5, instead if it is done following Theorem 3 we obtain a collection of INS-F.

In Figure 5 we represent only the structure of the NS-M, but every set can contain some elements such as XML files encoding the content of DOs. By means of NESTOR we separate the content from the hierarchical structure of DOs enabling a flexible way to access and exchange DOs in a distributed environment. In a distributed environment where two or more DLs exchange DOs we have to consider not only how to exchange the DOs but also which additional information may be necessary to properly understand the meaning of the exchanged DOs. For instance, if we consider the annotation $A_i \in H_{da}^{''}$, in order to understand its meaning we need, at least, to access the document annotated by A_i. In order to infer the context of A_i we need the DOs in the path from A_i to the root

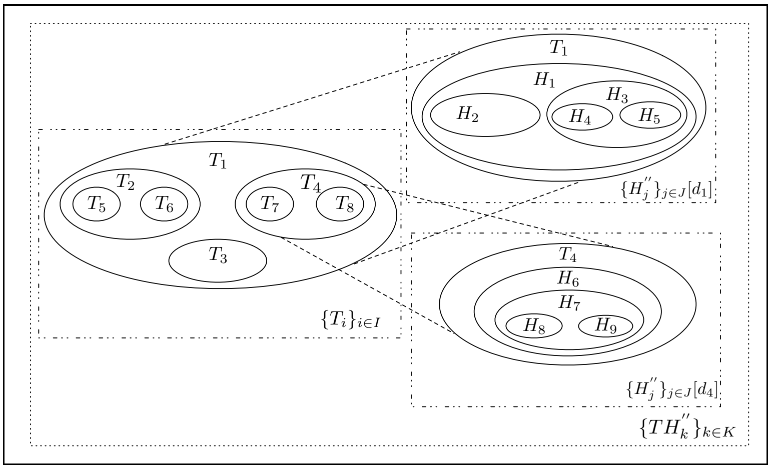


Fig. 5. The trees in Figure 4 mapped into NS-Families

of the tree $H^{''}_{da}$. The information brought by the root of the annotation tree could be not sufficient to understand the proper meaning of A_i, so we may have the necessity to reconstruct the path from A_i up to the root of the document tree $T(D_T, E_T)$. By means of NESTOR these operations can be easily done by mapping the trees into the INS-M that fosters the reconstruction of the upper levels of the hierarchy. Then the reconstruction of the path becomes a series of set operations that do not involve the content of DOs (for instance the XML files encoding the DOs). If we consider the INS-F $\{TH^{''}_k\}_{k \in K}$ we can determine the nearest common ancestor of two or more annotations simply intersecting the sets at which they belong; indeed, suppose we want to know the correlation between two annotations belonging to the same DO tree, by means of the INS-M we can determine their nearest common document throughout a single intersection operation between sets. On the other hand, we can decide to use the NS-M because it is useful to determine the descendants of a node. If we consider the NS-F $\{T_i\}_{i \in I}$, starting from a document we can reconstruct all its descendants in the document space, if we consider the collection of families $\mathcal{H}^{''}$ we can reconstruct all the annotations related to a document or related to a particular annotation and if we consider the NS-F $TH^{''}(DO_{TH^{''}}, E_{TH^{''}})$ starting from a document we can easily reconstruct all its descendants both in the document and in the annotation spaces. Furthermore, the use of set data models permits us to change the structure of the hierarchy without affecting the DOs content; indeed any variation in the hierarchical structure will affect only the inclusion order between the sets and not the data that is encoded into the XML files.

5 Final Remarks

In this paper we have described how the NESTOR framework can be applied to the FAST system enabling a set-theoretical view of annotation hierarchies. We have shown that documents and annotations are often organized into tree data structures and how these hierarchies can be joined together in a tree representing the whole information space composed by documents and annotations. Furthermore, we have pointed out some of the advantages of using the set data models defined in the NESTOR framework.

Future works will concern the definition of how to exchange annotations between DLs in a distributed environment exploiting the set-theoretical extension of the *Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)* described in [6]. Furthermore, we shall analyze how the application of NESTOR affects the FAST search framework [5]. From the text-based search point-of-view, the NESTOR Framework can be exploited to display the search results in the right context of the hierarchy and not only in a flat ranked list.

Acknowledgments

The work reported has been supported by a grant from the Italian Veneto Region. The study is also partially supported by EuropeanaConnect (Contract ECP-2008-DILI-52800).

References

1. Agosti, M., Ferro, N.: A formal model of annotations of digital content. ACM Trans. Inf. Syst. 26(1) (2007)
2. Candela, L., Castelli, D., Ferro, N., Koutrika, G., Meghini, C., Pagano, P., Ross, S., Soergel, D., Agosti, M., Dobreva, M., Katifori, V., Schuldt, H.: The DELOS Digital Library Reference Model. Foundations for Digital Libraries. ISTI-CNR at Gruppo ALI, Pisa (November 2007)
3. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press, Cambridge (2002)
4. Duranti, L.: Diplomatics: New Uses for an Old Science. Society of American Archivists and Association of Canadian Archivists in association with Scarecrow Press (1998)
5. Ferro, N.: Annotation Search: The FAST Way. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 15–26. Springer, Heidelberg (2009)
6. Ferro, N., Silvello, G.: The NESTOR Framework: How to Handle Hierarchical Data Structures. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 215–226. Springer, Heidelberg (2009)
7. Haslhofer, B., Jochum, W., King, R., Sadilek, C., Schellner, K.: The LEMO Annotation Framework: Weaving Multimedia Annotations With the Web. International Journal on Digital Libraries 10(1), 15–32 (2009)
8. Ioannidis, Y.E., Maier, D., Abiteboul, S., Buneman, P., Davidson, S.B., Fox, E.A., Halevy, A.Y., Knoblock, C.A., Rabitti, F., Schek, H.J., Weikum, G.: Digital Library Information-Technology Infrastructures. International Journal on Digital Libraries 5(4), 266–274 (2005)
9. Jech, T.: Set Theory-The Third Millenium edn. Springer, Heidelberg (2003)
10. Pitti, D.V.: Encoded Archival Description. An Introduction and Overview. D-Lib Magazine 5(11) (1999)
11. Vegas, J., Crestani, F., de la Fuente, P.: Context Representation for Web Search Results. Journal of Information Science 33(1), 77–94 (2007)