

Authoring Composite Documents and Their Descriptions

Nicolas Spyratos and Tsuyoshi Sugibuchi^(✉)

Laboratoire de Recherche en Informatique, Université Paris-Sud 11, Orsay, France
Nicolas.Spyratos@lri.fr,
tsuyoshi.sugibuchi@internetmemory.net

Abstract. We present a method for describing composite documents based on the descriptions of their components. Our main objective is to assist authors of composite documents in selecting documents and their descriptions during the authoring process. We assume that a document description is a set of terms from a given taxonomy, such that no two terms in the set are comparable. We call such a description a “reduced description” and we show that the set of all reduced descriptions forms a complete lattice under an appropriate ordering. Based on this lattice we introduce the concept of “admissible description” and we argue that admissible descriptions are the only ones that describe composite documents in a useful and meaningful manner.

1 Introduction

Today’s growth of digital publishing is bringing about not only media migration from atom to bit, but also more flexibility in authoring and customizing digital documents *after* their publication. For example, several non-profit projects and commercial companies start to offer *open textbook* platforms that intend to allow textbook authors, educators and students to create and customize textbooks. An interesting example is the *Connexions project* [1] funded by Rice University. In the Connexions’ repository, every textbook is managed as a collection of individual learning objects called *modules*. The Connexions’ website allows users not only to read textbooks but also to create and customize textbooks by composing modules taken from a variety of existing textbooks.

To make a new textbook by composing fragments of existing textbooks, authors need to find appropriate fragments from textbook repositories. At present, most open textbook platforms adopt description based document management. In such systems, each document and its fragments are associated with *descriptions*, also called *metadata*, two terms that we shall use interchangeably in the rest of the paper. Usually metadata contains free-text information including title, short description and free keywords, and information based on controlled vocabularies, or *taxonomies*, including subject category, topic group, etc. We note here that when the terms of a controlled vocabulary are hierarchically organized, then the controlled vocabulary is usually called a *taxonomy*.

Information based on controlled vocabularies is useful for more accurate and intelligent content retrieval, if metadata is properly created and maintained.

If we intend to allow users to take fragments from textbooks with smaller granularity, the cost of authoring metadata for each textbook fragment might be a problem. In particular, if authors have 100 % freedom of selection of terms for metadata, it rather makes metadata authoring tasks more difficult because authors need to choose terms without any clues. However, if we can define formal criteria of “better” or “consistent” descriptions, the story becomes slightly different. In this case, authoring systems can automatically check the current descriptions and give “suggestions” to authors to improve the descriptions. The main goal of this study is twofold: (a) to capture, and state formally, some simple but practically important requirements of descriptions, and (b) to demonstrate how to use them to help authors make good descriptions with less effort.

To this end, we propose a simple metadata management model for document composition environments. Our model assumes that (a) composite documents are structured as trees, whose nodes are either atomic documents, or other composite documents and (b) descriptions of documents are sets of terms taken from a taxonomy. The model does not consider contents of documents but deals only with their composition structure and the descriptions associated with the components of a composite document in order to *infer* appropriate descriptions (for the composite document).

Our basic assumption is that a document description is a set of terms from a given taxonomy, such that no two terms in the set are comparable. We call such a description a “reduced description”. In adopting this definition, our goal is to ensure that documents are described in a non redundant way and that, consequently, they can be retrieved more efficiently. We show that the set of all reduced descriptions forms a complete lattice under an appropriate ordering. Based on this lattice we introduce the concept of “admissible description” and we argue that admissible descriptions are the only (reduced) descriptions that describe composite documents in a useful and meaningful manner. We also outline an interactive process to assist users in authoring composite documents and choosing a desirable admissible description. We emphasize that the ultimate choice of a description is up to the document author and that the work presented here aims simply to assist the author in making an informed choice.

In the rest of the paper we first review some related studies (Sect. 2). Then we present our model for documents and their descriptions, as well as algorithms for description inference (Sect. 3). Based on this model, we introduce the concept of “admissible description” and outline an interactive process to assist users in authoring composite documents and choosing a desirable admissible description (Sect. 4).

2 Related Work and Preliminary Concepts

A lot of efforts have been devoted recently to develop languages and tools to generate, store and query metadata. Some of the most noticeable achievements

are the RDF language, RDF schema and several standards for representing controlled vocabularies, including OWL [4] and SKOS [5]. By using such languages and standards, several controlled vocabularies for metadata have been developed and are widely used in practice. These vocabularies include the ACM Computing Classification System [6] (for computer science), Gene Ontology [7] (genomics), AAT [8] (arts and architectures), DBPedia Ontology [9] (cross-domain ontology) and others. Most of these vocabularies are structured as general graphs including cycles. Even then most of these vocabularies also include hierarchically organized “is-a” relationships of terms.

In this paper, we focus on taxonomy-based descriptions [10], where a description is seen as a set of terms from a given taxonomy. The creation of such descriptions still remains mostly a manual process, possibly supported by acquisition software (for instance [11]). Usually, such description supports are performed by text analysis techniques (see for instance [12]) and some researches deal with *description propagation* to infer descriptions of derived contents from those of the original contents [13, 14].

The work in [3] which is the basis of our study also proposes a description inference model for composite documents. The description inference model proposed by [3] is mainly intended for document repository management. Based on this model, we proposed a framework for document description authoring, focusing in particular on description creation and description modification [2]. However, our previous work offers no clear discussion of what the requirements for an admissible description should be in order to support an on-line document ecosystem.

This paper is an extension of our previous work and its main goal is to focus on criteria for descriptions to be *admissible*, in the sense that they preserve integrity of document databases and provide enough information to cover document contents.

3 The Model of Composite Documents and Their Descriptions

3.1 Documents and Composite Documents

First of all, our model does *not* consider contents of documents. Our model deals only with structures of document composition and document descriptions. Therefore, we focus only on a document representation consisting of an identifier and a set of *parts*, as this is sufficient for our description management. Therefore, hereafter, when we talk of a document we shall actually mean its representation by an identifier and a set of parts. In order to define a document formally, we assume the existence of a countably infinite set *Doc* whose elements are used by all authors for identifying the created documents. For example, the set *Doc* could be the set of all URIs. In fact, we assume that the creation of a document is tantamount to choosing a (new) element from *Doc* and associating it with a set of other document identifiers that we call its parts.

Definition 1 (The representation of a document). A document consists of an identifier d together with a set of identifiers different than d , called the *parts* of d and denoted as $\text{parts}(d)$. If $\text{parts}(d) = \emptyset$ then d is called *atomic*, else it is called *composite*.

For notational convenience, we shall often write $d = d_1 + d_2 + \dots + d_n$ to stand for $\text{parts}(d) = \{d_1, d_2, \dots, d_n\}$.

In this paper, we assume that the structure of every composite document d is a tree in which d is the root, all atomic documents are leaves, and composite documents other than roots are intermediate nodes. Our choice is justified by the fact that (1) the tree is the most suitable structure for representing traditional books that are hierarchically organized, and (2) the tree is also a common structure adopted by many existing document composition environments including open textbook platforms. Based on this assumption, given a composite document d , each part d' of d is called a *child* of d , and d is called the *parent* of d' , denoted as $\text{parent}(d')$.

It is important to note that in our model the ordering of parts in a composite document is ignored because it is not relevant to our purposes. As we shall see shortly, deriving the description of a composite document from the descriptions of its parts does not depend on any ordering of the parts.

Note that a change in the structure of a composite document may require the use of new identifiers. For example, consider composite document d_0 of Fig. 1 in which the parts d_1 , d_2 and d_3 are at the same level. If the author of this composite document decides to group together d_1 and d_2 then it is necessary to introduce a new identifier, say d , to represent this grouping as a composite document. The new structure is shown in Fig. 1.

3.2 Taxonomy-Based Descriptions

Informally, a description in our model is just a set of terms taken from a given taxonomy. We would like to start our explanation about descriptions from the formal definition of taxonomy in our model.

Definition 2 (Taxonomy). Let T be a set of keywords, or *terms*. A *taxonomy* \mathcal{T} defined over T is a pair (T, \preceq) where \preceq is a reflexive and transitive binary relation over T , called *subsumption relation*.

Given two terms, s and t , if $s \preceq t$ then we say that s is *subsumed* by t , or that t *subsumes* s ; we also say that s is a *specialization* of t , or that t is a

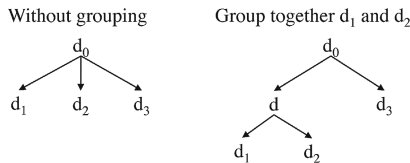


Fig. 1. Grouping of atomic documents

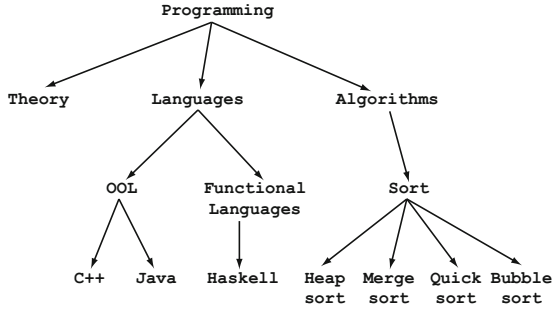


Fig. 2. A taxonomy

generalization of s . In our work, we assume that every taxonomy (T, \preceq) is a tree in which the nodes are the terms of T and where there is an arrow $s \rightarrow t$ iff s subsumes t in \preceq . Figure 2 shows the example taxonomy T_p that we use in this paper. In this example, the term **Sort** subsumes the term **Quick sort**, **OOL** subsumes **Java** and **C++**. Due to the transitivity of the subsumption relation, the term **Programming** subsumes all terms in the tree including itself.

In order to make a document sharable, a description of its content must be provided, so that users can judge whether the document in question matches their needs. Our model allows any set of terms from a taxonomy to be a description.

Definition 3 (Description). Given taxonomy (T, \preceq) , we call *description* in T any set of terms from T .

We end this section with an important remark concerning descriptions in general. In reality, a document has several attributes such as author name, title, creation date, and so on. So to describe a document one has to give one value per attribute. Therefore a description of a document should actually be a set of attribute-value pairs. For example the following is a description of a well known book by Agatha Christie: $\{(\text{Author}, \text{Agatha Christie}), (\text{Title}, \text{And then there were none}), (\text{Year}, 1939), (\text{Content description}, \text{novel})\}$. Note that the description can contain more than one value per attribute (e.g. if the document has more than one author).

The attribute values in such a description usually come from a controlled vocabulary and might be structured in a taxonomy (as is the case for the attribute **Content description** in our example). However, in order to simplify the presentation, we have assumed only one attribute for describing documents in this paper, namely **Content description**. As a consequence, we omit the attribute name in the description of a document and give only the set of attribute values. Moreover we have assumed that the values of this unique attribute are organized in a taxonomy.

Extending our results to descriptions involving more than one attribute is rather straightforward: instead of having single attribute values we will have tuples of values, and what needs to be done then is to define a subsumption

relation over tuples based on the subsumption relations over the individual attribute values. This extension lies outside the scope of the present paper and will be reported in future work.

As a final remark, note that the definition of a description as a set of attribute-value pairs can be represented in RDF in a straightforward way. Indeed, each pair $\{(\text{Attr}, \text{Value})\}$ can be represented by the RDF triple $\{(d, \text{Attr}, \text{Value})\}$, where d is the identifier of the document being described. This means that the document repository can then use the powerful reasoning system of RDF in order to handle description inferences; and moreover, the document repository can then communicate with other repositories using RDF, and in particular it can communicate with the world wide web. Similar advantages can be obtained using any other standard vocabulary of the Linked Open World (LOD). Indeed, the framework that we propose here works with any kind of controlled vocabulary. In fact, this ease to map to widely-used controlled vocabularies, gives a document repository based on our model enhanced visibility, searchability, and reusability of documents in the context of the world-wide-web. In a nutshell, descriptions in our model are easy to define and easy to map to RDF (with all the advantages entailed by such a mapping).

3.3 Inferred Descriptions

Reduction of a Description. A description can be redundant if some of its terms are subsumed by other terms in the description. For instance, the description $\{\text{Sorting}, \text{Quick sort}, \text{java}\}$ is redundant, as **Sorting** subsumes **Quick sort**. Redundant descriptions are sometimes undesirable as they can lead to redundant computations. Therefore we introduce the concept of non-redundant, or *reduced description*, defined as follows:

Definition 4 (Reduced description). Given taxonomy (T, \preceq) , a set of terms D from T is called *reduced* if for any terms s and t in D , $s \not\preceq t$ and $t \not\preceq s$.

In general, from the same redundant description we can derive multiple non-redundant descriptions, by either keeping only its maximal terms, or by keeping only its minimal terms. In this paper we consider both types of non redundant descriptions. Hence the following definitions:

Definition 5 (Cover of a description). Let D be a description in taxonomy (T, \preceq) . we call *cover* of D , denoted as $\text{cover}(D)$, the set of maximal terms in D .

Definition 6 (Reduction of a description). Let D be a description in taxonomy (T, \preceq) . we call *reduction* of D , denoted as $\text{red}(D)$, the set of minimal terms in D .

In our previous example, where $D = \{\text{Sorting}, \text{Quicksort}, \text{java}\}$, we have: $\text{cover}(D) = \{\text{Sorting}, \text{java}\}$ and $\text{red}(D) = \{\text{Quicksort}, \text{java}\}$. Note that every term in D is subsumed by some term in the cover and that every term in the cover subsumes some term in D .

Algorithm cover

Input a description D
Output $cover(D)$

```

 $C \leftarrow \emptyset$ 
for all  $t \in D$  do
  if there is no  $t' \in D$  such that  $t' \succeq t$  then
     $C \leftarrow C \cup \{t\}$ 
  end if
end for
return  $C$ 

```

Algorithm reduction

Input a description D
Output $red(D)$

```

 $R \leftarrow \emptyset$ 
for all  $t \in D$  do
  if there is no  $t' \in D$  such that  $t' \preceq t$  then
     $R \leftarrow R \cup \{t\}$ 
  end if
end for
return  $R$ 

```

Algorithm summary

Input a document d
Output $summary(d)$

```

if  $d$  is atomic then
  return  $cover(ADescr(d))$ 
end if
for all  $d_i \in parts(d), i = 1, \dots, n$  do
   $D_i \leftarrow summary(d_i)$ 
end for
 $P \leftarrow D_1 \times D_2 \times \dots \times D_n$ 
for all  $L_k = [t_1^k, \dots, t_n^k] \in P, k = 1, \dots, l$  do
   $T_k \leftarrow lub_{\preceq}(t_1^k, \dots, t_n^k)$ 
end for
return  $red(T_1, \dots, T_l)$ 

```

$lub_{\preceq}(t_1, \dots, t_n)$ returns the least upper bound of the set of terms t_1, \dots, t_n with respect to \preceq .

Fig. 3. cover, reduction and summary algorithm

Also note that every term in the description subsumes a term in the reduction, and there is no term in the reduction that is NOT subsumed by a term in the description. In other words, the reduction is the most compact and accurate representation of a redundant description, for searching purposes. For example, if a user wants to find documents describing something related to sorting, in the above example we can find the term **Sorting** in both, the $cover(D)$ and the reduction $red(D)$. But if a user wants to find documents related to “Quick sort”, the term **Quick sort** appears only in $red(D)$ and not in $cover(D)$. So $red(D)$ keeps more accurate terms from the original redundant description than $cover(D)$ does. Therefore $red(d)$ is more accurate than $cover(D)$.

The algorithms for computing the cover and the reduction of a description are illustrated in Fig. 3.

We note here that the cover, as well as other related concepts that we shall introduce shortly are used only internally to generate all admissible descriptions. It is up to the author to choose one among the admissible descriptions proposed by the authoring system. The chosen description then becomes *the* document description, and it is stored (along with the document identifier) in the document repository.

Based on the concepts of reduction and cover of a description, we now define some additional concepts to be used in the definition of admissible description.

Cover of a Document. Intuitively, the description of a composite document must incorporate somehow the descriptions of the document’s parts (and therefore, recursively, the descriptions of the document’s components). To state this

intuition formally, we have to extend the concept of cover from a single description to a set of descriptions.

Definition 7 (Cover of a document). Let $d = d_1 + \dots + d_n$, be a document with part descriptions D_1, \dots, D_n , respectively. The *cover* of d , denoted as $cover(d)$, is a description defined as $cover(d) = cover(D_1 \cup \dots \cup D_n)$.

For example, if d has two parts with descriptions $D_1 = \{\text{Sorting, Quick sort, java}\}$ and $D_2 = \{\text{Bubble sort, C++}\}$ then $cover(d) = \{\text{Sorting, java, C++}\}$. Note that the description obtained by simply taking the union of D_1 and D_2 gives an accurate but redundant description of the contents of the composite document.

Summary of a Document. Sometimes we want to summarize the topics contained in a big composite document. There are several possible approaches for summarization, and one of them is to use the document cover. Another approach is to extract common topics shared by all components of the document. For example, consider a composite document $d = d_1 + d_2$ with $\{d_1\} = \{\text{Quick sort, Java}\}$ and $\{d_2\} = \{\text{Bubble sort, C++}\}$. In this case, the description $D_{sum} = \{\text{Sort, OOL}\}$ is a possible summary of d_1 and d_2 . Indeed, **Sort** subsumes both **Quick sort** and **Bubble sort**, and **OOL** also subsumes both **Java** and **C++**. Therefore $\{\text{Sort, OOL}\}$ represents what d_1 and d_2 have in common.

In this example, $D'_{sum} = \{\text{Algorithms, Languages}\}$ is also a possible summary. However, D'_{sum} is less accurate than D_{sum} . The most extreme example is $D^*_{sum} = \{\text{Programming}\}$. D^*_{sum} summarizes any description in T but with the lowest accuracy. Usually such an over-general summary is useless for document search.

Now, intuitively, we can define the *summary of a document* as a description which (a) summarizes what all components of the document have in common in their descriptions and (b) it is minimal, in other words, has highest accuracy. The descriptions D'_{sum} and D^*_{sum} violate the second criterion because they have lower accuracy than D_{sum} .

In order to state this intuition formally, we introduce the following refinement relation over reduced descriptions.

Definition 8 (Refinement relation). Let D_1 and D_2 be two descriptions. We say that D_1 is *finer* than D_2 , denoted $D_1 \sqsubseteq D_2$, iff $\forall t_2 \in D_2, \exists t_1 \in D_1 \wedge t_1 \preceq t_2$.

For example, D_{sum} is finer than D'_{sum} , as for every term t in D'_{sum} , we can find a term in D_{sum} subsumed by t . Indeed, **Sort** \preceq **Algorithms** and **OOL** \preceq **Languages**.

The refinement relation \sqsubseteq is clearly reflexive and transitive. Moreover, over reduced descriptions, \sqsubseteq becomes antisymmetric. From these properties we can say that \sqsubseteq is a partial order over reduced descriptions (see [3] for more details). The following proposition states a more general property of the partial order \sqsubseteq , namely that it is a complete lattice. It is in this lattice that the summary of a composite document can be defined formally.

Proposition 1. The set of all reduced descriptions forms a complete lattice \mathcal{U} under the ordering \sqsubseteq . Moreover, for given a set $D = \{D_1, \dots, D_n\}$ of reduced descriptions, \mathcal{U} has least upper bound (lub), denoted as $\text{lub}(\mathcal{D}, \sqsubseteq)$ and greatest lower bound (glb), denoted as $\text{glb}(\mathcal{D}, \sqsubseteq)$.

The least upper bound of a set of descriptions is the most accurate set of terms representing what the descriptions have in common. Therefore, by obtaining the *lub* of descriptions of documents, we can get the most accurate description that summarizes what the documents have in common. By using this proposition, we can now define the summary of a document as follows:

Definition 9 (Summary of a document). Given a document d , the *summary* of d , denoted as $\text{sum}(d)$, is a description defined as follows:

- if d is atomic, $\text{sum}(d) = \text{cover}(D)$,
- else, for $d = d_1 + \dots + d_n$, let $\mathcal{D} = \{\text{sum}(d_1), \dots, \text{sum}(d_n)\}$, $\text{sum}(d) = \text{lub}(\mathcal{D}, \sqsubseteq)$.

The algorithm `summary` illustrated in Fig. 3 recursively computes the summary of a given document.

4 Admissible Descriptions

In this section, we would like to explain how we can use inferred descriptions of documents to help users to create and manage document descriptions. As we already mentioned, the author description of a document is left entirely up to description authors. Therefore, the algorithms explained in the previous section are not intended to *generate* descriptions of documents automatically. The role of the algorithms is to *suggest* inferred descriptions to avoid making descriptions from scratch. Before describing the principles underlying our suggestion process, we would like to introduce a basic concept, namely that of admissible description.

Admissible Descriptions. In order to define the concept of admissible description, the basic question is the following: are there any conditions that descriptions should satisfy in order to be admissible?

To begin with, intuitively, the description of a document should not be more general than the summary of the document. Consider for example a document d with two parts having the following descriptions: $D_1 = \{\text{Quick sort, Java}\}$, $D_2 = \{\text{Bubble sort, C++}\}$. The summary of d is then the following description: $\text{Sum}(d) = \{\text{Sorting, OOL}\}$. The description $\{\text{Computer Science}\}$ strictly subsumes the summary of d and is therefore too general to describe d . On the other hand, the descriptions $\{\text{Sorting, Java, C++}\}$ and $\{\text{Quick sort, Bubble sort, OOL}\}$ are both subsumed by the summary of d and therefore each of them can be used to describe d (and so can the summary itself).

Note that none of these two descriptions is better than the other (in fact they are not comparable); the description $\{\text{Sorting, Java, C++}\}$ would be used if

the author wanted to emphasize the language aspect of the content of d , while the description $\{\text{Quick sort, Bubble sort, OOL}\}$ would be used if the author wanted to emphasize the algorithmic aspect of the content of d .

However for a description to be *admissible*, it is not sufficient to just be subsumed by the summary of the document being described. Suppose for example that the descriptions of the two parts of d were the following: $D_1 = \{\text{Quick sort, Java}\}$, $D_2 = \{\text{Bubble sort, C++, Analytics}\}$. Then the summary remains the same: $\text{Sum}(d) = \{\text{Sorting, OOL}\}$. Consider now the following description: $D = \{\text{Quick sort, Java, Bubble sort, C++}\}$. This description is subsumed by the summary, yet it leaves out the term **Analytics** (i.e. it does not subsume the cover of d). On the other hand, if we remove the term **Analytics** from D_2 and add it to D then D is still subsumed by the summary but the term **Analytics** in description D of d does not appear in the descriptions of the components of d . In other words, D is simply a “bad” description because it describes something which does not appear in the descriptions of its parts.

To summarize our discussion so far, a description D should be subsumed by the summary, and moreover it should be *sound* and *complete* in order to be admissible. By soundness we mean that each term of D should subsume some term of the cover; and by completeness we mean that each term in the cover should be subsumed by some term of D . These requirements are stated formally in the following definition.

Definition 10 (Admissible description). *Let $\text{Cover}(d)$ and $\text{Sum}(d)$ be the cover and the summary of a document d . A reduced description D is called an admissible description of d if the following conditions hold:*

- $D \sqsubseteq \text{Sum}(d)$
- for each term t in D there is a term s in $\text{cover}(d)$ such that $s \preceq t$ (soundness)
- for each term s in $\text{cover}(d)$ there is a term t in D such that $s \preceq t$ (completeness)

Soundness of descriptions is an indispensable property for every description in order to preserve integrity of document databases. If a description of a composite document doesn’t satisfy soundness, it means that the description contains a term which is not in the description of any component of the document. Documents with non-sound descriptions may appear as non-relevant documents in document search results.

On the other hand, completeness comes from a more practical requirement, namely the requirement of “minimal surprise”. Intuitively, this requirement is satisfied if every term in a component description or a generalization thereof appears in the description of its parent. This constraint minimizes the risk that a reader meets unexpected contents in a document (i.e. contents not mentioned in the document description).

Note that the cover is always an admissible description. The summary on the other hand is not always an admissible description, though it is always a sound description.

Clearly, in a composite document with a big number of components the number of admissible descriptions might be quite large, and choosing one among many admissible descriptions becomes a rather tedious task. Hence the need for a user friendly interface helping the user to choose, in a systematic manner, one among possibly many admissible descriptions. In the remaining of this section we outline some basic principles that such an *authoring system* should follow.

The scenario that we consider is that of an author composing a document and having at least a fair knowledge of the subject area of the documents being composed (e.g. computer science, literature etc.) as well as of the taxonomy being used. The author has two options: either to ask the authoring system to suggest an admissible description for the composite document or to submit his own description to the authoring system for approval. In both cases the authoring system uses the inference mechanisms that we saw earlier in order to generate admissible descriptions or to check proposed descriptions for admissibility.

In the first case, the authoring system will generate all admissible descriptions of the composite document and will present them to the author so that he can choose one. In the second case, the authoring system will check whether the description proposed by the author is an admissible description; if yes, then the description is accepted, otherwise an interaction takes place between the author and the authoring system. During this interaction, the authoring system helps the author to modify his description so that it becomes admissible.

If the proposed description is found to be non-admissible, this means that it does not satisfy soundness or completeness. To see what kind of interaction is taking place in this case, let $ADescr(d)$ denote the description proposed by the author for a composite document d , and suppose that $d = d_1 + d_2$. Moreover, suppose that $ADescr(d_1) = \{\text{Quick sort, Java}\}$, $ADescr(d_2) = \{\text{Bubble sort, C++}\}$ and $ADescr(d) = \{\text{Sort, Theory}\}$. This description does not satisfy soundness because none of the terms in d_1 and d_2 is subsumed by **Theory** in $ADescr(d)$. In this case, the authoring system can suggest the following two options:

- Remove terms: the system suggests to remove **Theory** from $ADescr(d)$.
- Add components: the system shows to the user a list of components whose descriptions contain a term $t \preceq \text{Theory}$ and suggests the addition of one of them.

On the other hand, if a description does not satisfy completeness, then the system can again suggest two options. For example, the author description $ADescr(d) = \{\text{Bubble sort, OOL}\}$, does not satisfy completeness due to lack of mention about **Quick sort** in $ADescr(d_1)$. A reader who intends to learn about bubble sort may be surprised when he faces a description of quick sort which is not mentioned in the description. In this case, the author has the following two options:

- Remove components: the system suggests to remove d_1 from the composite document.

- Add or generalize terms: the system shows a list of terms $\{t\}$ satisfying $t \succeq \text{Quick sort}$ and suggests the addition of one of them, or generalizes a term in $ADescr(d)$ by one of them. For instance, an author can add **Quick sort** to $ADescr(d)$, or replace **Bubble sort** in $ADescr(d)$ by **Sort** which subsumes both **Quick sort** and **Bubble sort**.

Such a description improvement process is usually an interactive process because modification of a description in order to satisfy one property may break another property. For each modification made by authors the authoring system should check soundness and completeness and suggest next options if a modified description does not satisfy these properties.

5 Concluding Remarks

We have seen a method for describing composite documents based on the descriptions of their components. We defined an ordering on reduced descriptions and gave algorithms for inferring descriptions based on that ordering. We also used the description ordering to introduce the concept of “admissible description”; and argued that admissible descriptions are the most appropriate for composite documents. However, as the number of such descriptions might be large, we discussed an interactive approach allowing authors of composite documents to build descriptions incrementally with the help of an authoring system.

The important features of descriptions as defined in this paper is that they are easy to use and easy to map to RDF. This means that document repositories based on our model have enhanced visibility, searchability, and reusability of their documents in the context of the world wide web.

Our current work focuses on two topics:

- The extension of the model to handle descriptions over two or more attributes (not just over the attribute Content Description). As we mentioned in the paper, such an extension is possible and involves mainly the extension of the ordering from descriptions over the values of a single attribute to descriptions over tuples of values defined on two or more attributes.
- The design of an authoring system using the basic principles discussed in the paper.

References

1. Connexions web site. <http://cnx.org/>
2. Sugibuchi, T., Tuan, L.A., Spyratos, N.: Metadata inference for description authoring in a document composition environment. In: Agosti, M., Esposito, F., Ferilli, S., Ferro, N. (eds.) IRCDL 2012. CCIS, vol. 354, pp. 69–80. Springer, Heidelberg (2013)
3. Rigaux, P., Spyratos, N.: Metadata inference for document retrieval in a distributed repository. In: Maher, M.J. (ed.) ASIAN 2004. LNCS, vol. 3321, pp. 418–436. Springer, Heidelberg (2004)

4. OWL 2 Web Ontology Language Document Overview. <http://www.w3.org/TR/owl2-overview/>
5. SKOS Simple Knowledge Organization System Reference. <http://www.w3.org/TR/skos-reference/>
6. Coulter, N.: ACM's computing classification system reflects changing times. *Commun. ACM* **40**(12), 111–112 (1997)
7. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat. Genet.* **25**(1), 25–29 (2000)
8. AAT Web site. <http://www.getty.edu/research/tools/vocabularies/aat/>
9. The DBPedia Ontology. <http://wiki.dbpedia.org/Ontology>
10. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): *Modern Information Retrieval*. Addison-Wesley, Boston (1999)
11. Erdmann, M., Maedche, A., Schnurr, H.-P., Staab, S.: From manual to semi-automatic semantic annotation: about ontology-based text annotation tools. In: *Proceedings of the COLING International Workshop on Semantic Annotation and Intelligent Context* (2000)
12. Handschuh, S., Staab, S., Volz, R.: On deep annotation. In: *Proceedings of International World Wide Web Conference (WWW)*, pp. 431–438 (2003)
13. Pastorello Jr, G.Z., Daltio, J., Medeiros, C.B.: Multimedia semantic annotation propagation. In: *Proceedings of IEEE International Symposium on Multimedia (ISM)* 08, pp. 509–514 (2008)
14. Leung, M.-K., Mandl, T., Lee, E.A., Latronico, E., Shelton, C., Tripakis, S., Lickly, B.: Scalable semantic annotation using lattice-based ontologies. In: Schürr, A., Selic, B. (eds.) *MODELS 2009. LNCS*, vol. 5795, pp. 393–407. Springer, Heidelberg (2009)

Information Search, Integration, and Personalization
International Workshop, ISIP 2013, Bangkok, Thailand,
September 16--18, 2013. Revised Selected Papers
Kawtrakul, A.; Laurent, D.; Spyratos, N.; Tanaka, Y.
(Eds.)
2014, X, 137 p. 51 illus., Softcover
ISBN: 978-3-319-08731-3