Metadata Inference for Description Authoring in a Document Composition Environment

Tsuyoshi Sugibuchi, Ly Anh Tuan, and Nicolas Spyratos

Laboratoire de Recherche en Informatique, Université Paris-Sud 11, France
{Tsuyoshi.Sugibuchi,Anh_Tuan.Ly,Nicolas.Spyratos}@lri.fr

Abstract. In this paper, we propose a simple model for metadata management in a document composition environment. Our model considers (1) composite documents in the form of trees, whose nodes are either atomic documents, or other composite documents, and (2) metadata or *descriptions* of documents in the form of sets of terms taken from a taxonomy. We present a formal definition of our model and several concepts of inferred descriptions. Inferred descriptions can be used for *term suggestion* that allows users to easily define and manage document descriptions by taking into account what we call *soundness* of descriptions.

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 their descriptions, also called *metadata*. 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. 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 many metadata for each textbook fragment might be a problem. A clue for reducing such metadata authoring cost is the fact that each fragment of a textbook is usually part of a bigger context. For instance, a section of a textbook usually has a previous or next siblings, and a parent chapter that encloses child sections. By taking into account such relationships among fragments, we can infer metadata of new fragments from the metadata of existing fragments. Metadata of textbook fragments should be manually made by human-beings, but machines can also "suggest" inferred metadata. Such metadata suggestions will help users in easily making metadata.

In this paper, we propose a simple metadata management model for document composition environments. Our work is based on the metadata inference model for composite documents proposed in [2]. The model described in [2] mainly focuses on document *sharing*. In the present paper, we focus on the actual usage of metadata for *authoring* document descriptions. We give the formal definition of our metadata model and demonstrate how it can be used to suggest terms for descriptions based on descriptions of existing documents.

In the rest of this paper we first review some related studies (Section 2). Then we describe our metadata model and some algorithms for inferring metadata (Section 3). Based on this model, we introduce a criterion that every description should satisfy, and then we explain how our "term suggestion" by using this criterion (Section 4).

2 Related Work

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 schemas and several standards for representing controlled vocabulary including OWL [3] and SKOS [4]. By using such languages and standards, several controlled vocabularies for metadata have been developed and are widely used in practice. These vocabularies include Gene Ontology [5] (genomics), AAT [6] (arts and architectures), DBPedia Ontology [7] (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 annotations [8] to describe the content by using such hierarchically organized sets of terms. Generation of such annotations still remains mostly a manual process, possibly supported by acquisition software (for instance [9]). Many of such annotation supports are performed by text analysis techniques (for instance [10]) and some researches deal with *annotation propagation* to infer metadata of derived contents from those of the original based content authoring processes [11][12]. The work in [2] which is the basis of our study also proposes a metadata inference model for composite documents. However, the inference model of [2] is mainly intended for document repository management. In contrast, the inference model that we propose here is intended for document description authoring, including creation and modification.

3 The Model of Composite Documents and 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 metadata management. Therefore, hereafter, when we talk of a document we shall actually mean its representation by an identifier and a set of parts.

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

For notational convenience, we shall often write $d = d_1 + d_2 + \ldots + d_n$ to stand for $parts(d) = \{d_1, d_2, \ldots, d_n\}$. Based on the concept of parts, we can now define the concept of *component.*

Definition 2 (Components of a document). Let $d = d_1 + d_2 + \ldots + d_n$. The set of *components* of d, denoted as $comp(d)$, is defined recursively as follows:

– if d is atomic, then $comp(d) = \emptyset$
– else $comp(d) = parts(d) \cup comp(d_1) \cup comp(d_2) \cup \ldots \cup comp(d_n)$.

In this paper, we assume that every composite document d is a tree in which d is the root and $comp(d)$ is the set of 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, for a composite document d and its part $d' \in parts(d)$, d' is called *child* of d, and d is called *parent* of d', denoted as $parent(d')$. 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.

3.2 Taxonomy and Description

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

Definition 3 (Taxonomy). Let T be a set of keywords, or *terms.* A *taxonomy* $\mathcal{T}$ defined over T is a tuple $(T, \preceq)$ 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. In our work, we assumes 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

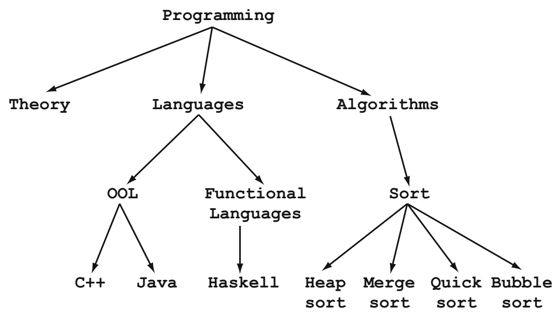

Fig. 1. A taxonomy

subsumes t in $\preceq$. Fig 1 shows the example taxonomy $\mathcal{T}_p$ 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 the rest of this paper, we use a symbol $tail(t)$ that stands for the set of all terms in the taxonomy strictly subsumed by t, i.e., $tail(t) = \{s | s \prec t\}$.

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 sets of terms from a taxonomy as descriptions.

Definition 4 (Description). Given taxonomy $(T, \preceq)$, we call *description* in T any set of terms from T.

3.3 Inferred Descriptions

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

Definition 5 (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 $t \not\preceq s$.

Following the above definition, we can make a description non-redundant by either removing all but its minimal terms, or by removing all but its maximal terms. We shall assume the former as it produces more accurate descriptions. This should be clear from our previous example, where the description {`Quick sort, Java`} is more accurate than {`Sort, Java`}. Hence the following definition:

Definition 6 (Reduction). Given a description D in taxonomy $(T, \preceq)$, we call *reduction* of D, denoted as $reduce(D)$, the set of minimal terms in D with respect to the subsumption $\preceq$.

The important point to note is that even if a description created by an author contains redundancy, our model respects the original form of the description and does not remove anything from the description. The choice of terms to include in a description is left entirely up to description authors. Reduction of descriptions and other concepts of inferred descriptions are made only internally, or for suggesting "hints" for users to create descriptions with less effort.

Therefore we distinguish between descriptions created by authors and descriptions inferred automatically by using algorithms. For a document d, the former type of description is called the *author description* of d, denoted as $ADescr(d)$: author descriptions are exactly the descriptions that authors create and a document repository stores. The latter type of description is what is generated internally, by machines, for helping authors in description authoring.

Additionally, we would like to introduce two more concepts of inferred descriptions.

Cover of a Document. To make a description of a composite document, it is sometimes useful to know all topics covered by the components of the document. Now we introduce the concept of *cover* of a document d that is an inferred description formed with a minimum set of terms and semantically covers all terms appearing in descriptions of d's components. The cover of a document is formally defined as follows:

Definition 7 (Cover of a document). Given a document d, the *cover* of d, denoted as $cover(d)$, is a description recursively defined as follows:

- if d is atomic, $cover(d) = reduce(ADescr(d))$,
- else, for $d = d_1 + \ldots + d_n$, $cover(d) = reduce(cover(d_1) \cup \ldots \cup cover(d_n))$.

Informally, the cover of a document is the minimum but most accurate description of the document. To create the description of a document, authors should choose terms present in the cover of the document, or terms subsuming at least one term in the cover. Otherwise the created description might contain terms not related to any component of the described document.

Summary of a Document. On the other hand, sometimes we want to summarize topics of a big composite document. There are several possible approaches for summarization. One intuitive approach is to extract common topics shared by all components of a document. Suppose a composite document $d = d_1 + d_2$ such that $ADescr(d_1) = \{$`Quick sort, Java`$\}$ and $ADescr(d_2) = \{$`Bubble sort, C++`$\}$. In this case, $D_{sum} = \{$`Sort, OOL`$\}$ is a possible summary of d_1 and d_2. `Sort` subsumes both `Quick sort` and `Bubble sort`. `OOL` also subsumes both `Java` and `C++`. As the result, $\{$`Sort, OOL`$\}$ represents what d_1 and d_2 have in common.

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

Now we informally define the *summary of a document* as a description such that (1) it summarizes what all components of a document have in common in their descriptions and (2) it is minimal, in other words, has highest accuracy.

The examples of D'_{sum} and D^*_{sum} violate the second criterion because they have lower accuracy than D_{sum}.

In order to formalize this definition, we introduce the following *refinement relation on 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}, i.e., $D_{sum} \sqsubseteq D'_{sum}$ because for every term t in D'_{sum}, we can find a term in D_{sum} subsumed by t such as in our example, where `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 of $\sqsubseteq$, we can say that $\sqsubseteq$ is a partial order over reduced descriptions, and a set of reduced descriptions has a least upper bound in $\sqsubseteq$. Here we omit the detail and just introduce the following proposition and theorem. For detailed discussion and proofs of them, see [2].

Proposition 1. The relation $\sqsubseteq$ is a partial order over the set of all reduced descriptions.

Theorem 1. Let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be any set of reduced descriptions. Let $\mathcal{U}$ be the set of all reduced descriptions S such that $D_i \sqsubset S, i = 1, \ldots, n$, i.e., $\mathcal{U} = \{S | D_i \sqsubseteq S, i = 1, \ldots, n\}$. Then $\mathcal{U}$ has a least upper bound, that we shall denote as $lub(\mathcal{D}, \sqsubseteq)$.

The least upper bound (*lub*) 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 theorem, we can now define the summary of a document as following:

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

– *if d is atomic,* $summary(d) = reduce(ADescr(d))$*,*
– *else, for* $d = d_1 + \ldots + d_n$*, let* $\mathcal{D} = \{summary(d_1), \ldots, summary(d_n)\}$*,* $summary(d) = lub(\mathcal{D}, \sqsubseteq)$*.*

The algorithm `summary` illustrated in Fig. 2 recursively computes the summary of a given document. We shall use these algorithms in the next section for helping authors to make descriptions of new documents.

4 Metadata-Aided Suggestion in Document Description Authoring

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

Algorithm `summary`

```
Input a document $d$
Output $summary(d)$

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

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

Algorithm `cover`

```
Input a document $d$
Output $cover(d)$, or false if $d$ contains documents that have descriptions not satisfying soundness.

if $d$ is atomic then
   return $reduce(ADescr(d))$
end if
$C \leftarrow \emptyset$
for all $d' \in parts(d)$ do
   $C' \leftarrow$ cover($d'$)
   if $C'$ = false then return false end if
   $C \leftarrow C \cup C'$
end for
for all $t \in ADescr(d)$ do
   if there is no $t' \in C'$ such that $t' \preceq t$ then
      return false
   end if
end for
return $reduce(C)$
```

Fig. 2. `summary` algorithm and `cover` algorithm

to description authors. Therefore, the algorithms explained in the previous section are not intended to *generate* descriptions of documents automatically. The purpose of the algorithms is to *suggest* inferred descriptions to avoid making descriptions from scratch. Before entering into details of our suggestion process, we would like to discuss two preliminary topics.

Soundness of Descriptions. While authors can freely choose any terms to make descriptions, are there any criteria that descriptions should satisfy? Our opinion is that every description should satisfy some kind of "soundness". If a description contains a term t, the described document should contain something related to the term t. Now we formalize soundness of a description as follows:

Definition 10 (Soundness of a document description). A description D of document d is called *sound* if d and D satisfy the following condition:

– For every term $t \in D$, at least one term $t' \in cover(d)$ is subsumed by t, or
– d is atomic

We should comment on the last part of this definition. As we mentioned several times, our model does not deal with contents of documents. Consequently, our model has no way to determine whether an author description of an atomic document satisfies soundness or not with respect to the document content. Therefore, we firstly *believe* it. Our model depends on an assumption that all descriptions of atomic documents satisfy soundness.

In the rest of this paper, we would like to adopt the *soundness criterion* that requires every author description to be sound. We think it is a reasonable criterion for keeping integrity of a document repository. Without this criterion, a document repository might have an untrustworthy description that contains terms not related to any parts of a described document.

The soundness of a document description is defined over the cover of a document. Therefore, the algorithm `cover` illustrated in Fig. 2 can validate soundness of descriptions and compute cover of descriptions in parallel by using simple depth-first search. We also use a symbol $dtDom(d)$ that stands for the set of all terms in T such that we can use for describing a composite document d, i.e., $dtDom(d) = \{t | t \in T \wedge \exists t' \in cover(d) \wedge t' \preceq t\}$.

Types of Terms for Suggestion. Briefly, suggestion is an activity to indicate a *suggestion list* of choices to users for allowing them to easily specify input values. In this paper, we do not deal with details of user interaction design. Here we would like to just classify terms for suggestion into the following three different types of term sets by how terms are initially selected in a suggestion list and what users can do on terms in a suggestion list.

– $\boldsymbol{D_{rec}}$ **(recommended)**: All terms in this set are selected as default and users can remove terms from the set
– $\boldsymbol{D_{opt}}$ **(optional)**: All terms in this set are not selected as default and users can add terms to the set
– $\boldsymbol{D_{obso}}$ **(obsolete)**: All terms in this set are not selected as default and users cannot select any of them.

D_{rec} typically contains terms that affect summary of descriptions. By removing terms in D_{rec}, authors can *generalize* descriptions to give broader meanings. D_{opt} typically contains terms that do not affect the summary but can be used for descriptions. By selecting terms in D_{opt}, authors can *specialize* descriptions to give narrower meanings. In many cases, the size of D_{opt} is very big. Therefore, sometimes we partition D_{opt} into a sequence of disjoint subsets $D_{opt1}, D_{opt2}, \ldots$ by priorities of terms. D_{obso} contains terms that violate the soundness criterion of descriptions. D_{obso} is used for indicating what must be removed from descriptions to preserve soundness. Authors cannot change selections of terms in D_{obso}. Authors can only accept removing indicated terms from descriptions.

4.1 Create New Atomic Documents

When an author creates a new atomic document as an independent one, the author needs to choose terms to define its description by taking into account the document content. However, if an author writes a new atomic document as a part of an existing composite document, we can suggest terms for its description by taking into account the descriptions of existing components.

Let d_p be a composite "parent" document. Suppose an author has created a new atomic document d as a child of d_p and now he is going to define an author description $ADescr(d)$. Firstly, if a parent document has its summary, the same summary should also be a "sound" description of a new child document. In this case, the author should define an author description $ADescr(d)$ such that $ADescr(d) \sqsubseteq summary(d_p)$.

See the example illustrated in Fig. 3. In this example, the summary D_{sum} of d_p is {`Sort, Java`}. If the author defines $ADescr(d)$ as {`Sort, Java`} or {`Quick sort, Java`}, D_{sum} does not change. On the other hand, if the author defines $ADescr(d)$ as {`Sort`}, D_{sum} will be changed to {`Sort`} that has less accuracy. It is the absence of `Java` from $ADescr(d)$ that causes such change of the document summary. Therefore, even if the author wants to remove this

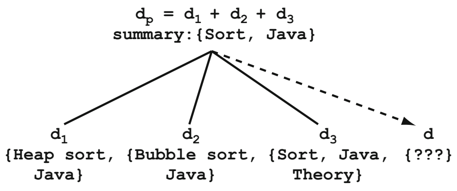

Fig. 3. Creation of an atomic document

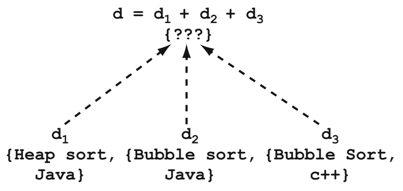

Fig. 4. Creation of a composite document

term from the description, it should be *intentionally* removed. Therefore, terms in D_{sum} should be suggested as members of D_{rec}. Because terms in D_{rec} are selected as default in a suggestion list, then users can *manually* remove them.

Secondly, a description of a new child document should be more specialized than the summary of its parent. We can perform such specialization (1) by specializing a part of the summary of a parent, or (2) by adding new terms in T. Regarding point (1), we can extract promising candidates of terms for specialization by comparing descriptions of *siblings*. Let's go back to the example in Fig. 3. In this example, siblings of d have $\{$`Heap sort, Java`$\}$, $\{$`Bubble sort, Java`$\}$, $\{$`Sort, Java, Theory`$\}$ as their descriptions. Taking $D_{sum} = \{\texttt{Sort}, \texttt{Java}\}$ into account, we can see that `Sort` is specialized in descriptions of some siblings but `Java` is not. Therefore, tails of `Sort`, for instance `Quick sort` and `Merge sort`, have higher priority for specialization than tails of `Java`. We use a symbol *specializedIn*(d) that stands for the set of all terms in *summary*(d) that are specialised in some descriptions of d's parts, i.e., $\mathit{specializedIn}(d) = \{t | t \in \mathit{summary}(d) \wedge \exists d' \in \mathit{part}(d) \wedge \exists t' \in \mathit{ADescr}(d') \wedge t' \prec t\}$.

Regarding point (2), any terms in T also can be candidates for specialization but have lower priority than the ones suggested in (1).

Summing up, for a new atomic document d which is a part of d_p, we can suggest the following sets of terms for a description of d.

- $D_{rec} = D_{sum} = \mathit{summary}(d_p)$
- $D_{opt1} = \{t | t_s \in \mathit{specializedIn}(d_p) \wedge t \in \mathit{tail}(t_s)\}/D_{rec}$
- $D_{opt2} = T/(D_{rec} \cup D_{opt1})$
- $D_{obso} = \emptyset$

In the above sets of terms, $D_{rec} \cup D_{opt1} \cup D_{opt2}$ is equal to T. This means that authors can choose any sets of terms from T as a description of an atomic document even if the atomic document is a part of a composite document.

4.2 Create New Composite Documents

On the other hand, to define a description of a composite document, there is a strict criterion the description should satisfy, namely soundness. Suppose an author has made a composite document d with its components *comp*(d) and now he is going to define an author description *ADescr*(d). In this case, any term in *ADescr*(d) should be a member of $\mathit{dtDom}(d) = \{t | t \in T \wedge \exists t' \in \mathit{cover}(d) \wedge t' \preceq t\}$ to satisfy the soundness. Therefore we use *dtDom*(d) as D_{opt} for suggestion.

Regarding D_{rec}, there is no criterion for determining terms that a description of a composite document should have. However, to construct a description from an empty set of terms is a troublesome task. Therefore, here we would like to use a summary of a document as the starting point of description authoring. $D_{sum} = summary(d)$ is suggested as D_{rec} for a description.

The important point to note is that, in this case, terms in D_{sum} are not mandatory for a description of d. The author can remove terms belonging to D_{sum} from $ADescr(d)$ to hide some contents included in d. For instance, in the example illustrated in Fig. 4, the summary of the composite document is {`Sort, OOL`}. However, if the description author thinks that programming languages are not in important topic in the context of the composite document, he can drop `OOL` and keep only {`Sort`} as a description of the composite document.

As a consequence, for a new composite document d, we can suggest the following sets of terms for a description of d.

- $D_{rec} = summary(d)$
- $D_{opt} = dtDom(d)/D_{rec}$
- $D_{obso} = \emptyset$

Additionally, when an author makes a new composite document as a part of an existing document, we can give more accurate suggestion by comparing with siblings of the new document. See the example illustrated in Fig. 5. In this example, originally the summary of $d = d_4 + d_5$ is just {`Quick sort`}. However, this description might be too brief because all siblings of d have terms related to programming languages in their descriptions. We can capture such topics shared by siblings as a summary of a parent document. In this example, the parent document $d_p = d_1 + d_2 + d_3$ has its summary {`Sort, Languages`}. Therefore, terms in $cover(d)$ related to, more precisely, subsumed by `Sort` or `Languages` also should be suggested as a part of a description of d. As a result, we get `C++` as an additional member of D_{rec}. Finally, we can suggest {`Quick sort, C++`} as an initial description of d.

To sum up, for a new composite document d which is a part of d_p, we can suggest the following sets of terms for a description of d.

- $D_{rec} = reduce(summary(d) \cup \{t | t \in cover(d) \wedge \exists t_s \in summary(d_p) \wedge t \preceq t_s\})$
- $D_{opt} = dtDom(d)/D_{rec}$
- $D_{obso} = \emptyset$

4.3 Removing Parts of Documents or Document Descriptions

When an author removes some parts of a composite document, or terms from descriptions of document components, such operation might change the cover of the document therefore it might affect soundness of the document description. To preserve soundness of descriptions, risk of soundness violation should be checked before applying operations and be appropriately notified with a list of terms that should be removed to keep soundness.

The algorithm `checkSoundness` in Fig. 6 takes a document d and a set of terms D to remove from $ADescr(d)$, or $cover(comp(d))$ when the document d

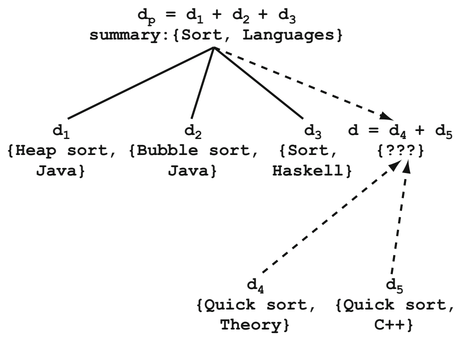

Fig. 5. Creation of a composite document as a part of an existing document

Algorithm `checkSoundness`

```
Input
A document $d$. A set of terms $D$ to remove, or
$D = cover(comp(d))$ when $d$ itself is removed.
Output
A set of mappings $M_v = \{m_v : d_v \mapsto D_v\}$ such
that $d_v$ is a document having a set of terms $D_v$
that violate the soundness criterion.

if $d$ has no parent then return $\emptyset$ end if
$d_p \leftarrow parent(d)$
$D_r \leftarrow cover(d)/D$
for all $d' \in parts(d_p)$ such that $d' \neq d$ do
    $D_r \leftarrow D_r \cup cover(d')$
end for
$D \leftarrow reduce(D_r \cup D)/D_r$
if $D = \emptyset$ then return $\emptyset$ end if
$M_v \leftarrow$ checkSoundness$(d_p, D)$
$D_v \leftarrow \{t | t \in ADescr(d_p) \wedge \exists t' \in D \wedge t \preceq t'\}$
if $D_v \neq \emptyset$ then $M_v \leftarrow M_v \cup \{d_p \mapsto D_v\}$ end if

return $M_v$
```

Fig. 6. `checkSoundness` algorithm

itself is removed. Then this algorithm returns a set of mappings $M_v = \{m_v : d_v \mapsto D_v\}$ such that d_v is a document having a set of terms D_v that violate the soundness criterion. This algorithm recursively propagates terms in D from a child document to its parent. In each propagation step, terms compensated by descriptions of sibling documents are removed from D.

By using this algorithm, we can notify authors about document descriptions affected by removing operations, and indicate suggestion lists of terms for updating descriptions to keep soundness. Suppose an author intends to remove a set of terms D from a description of d. In this case, firstly we need to compute a set of mappings $M_v = \texttt{checkSoundness}(d_p, D)$. Then for each d_v with a term set D_v in $M_v = \{m_v : d_v \mapsto D_v\}$, we can suggest the following sets of terms:

– $D_{rec} = ADescr(d_v)/D_v$
– $D_{opt} = \emptyset$
– $D_{obso} = D_v$

5 Concluding Remarks

In this paper, we have presented a model for metadata of composite documents. Our model allows authors to freely choose terms from a taxonomy to make descriptions. However, once documents are placed in composite documents, we can infer various restrictions and suggestions on terms for descriptions by taking into account soundness of descriptions. We think soundness of descriptions is a simple but essential criterion for keeping integrity of a document repository.

In future work, an urgent task is prototyping for identifying matching points and mismatch between the model we have proposed here and problems in practice. As we have seen in this paper, the modeling part of this study is very abstract. Currently we have a plan to prototype a document management system that uses this model in the metadata management part.

Regarding the model, firstly, we would like to extend the concept of document summary. In the current definition, a document summary summarizes topics shared by all *atomic documents*. However, summaries produced by using this definition are somehow too brief. Instead of summarizing at the level of atomic documents, we can summarize a document at a coarser granularity, for instance, direct children of a document to summarize, so that we can get more detailed summaries. In a future study we would like to introduce a *degree of summarization* to control the level of detail of summaries, and use it for assisting description authoring.

Finally, we have discussed only a case that a document has up to one parent. However, if a document is used as part of multiple composite documents, a document can have multiple parents. In this case, we can compare usage of the same document in different composite documents. We would like to find a way to use such comparison of document usage for improving term suggestion.

Acknowledgement. This work was partially supported by the European project ASSETS: Advanced Search Services and Enhanced Technological Solutions for the European Digital Library (CIP-ICT PSP-2009-3, Grant Agreement no 250527).

References

1. Connexions web site, `http://cnx.org/`
2. 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)
3. OWL 2 Web Ontology Language Document Overview, `http://www.w3.org/TR/owl2-overview/`
4. SKOS Simple Knowledge Organization System Reference, `http://www.w3.org/TR/skos-reference/`
5. The Gene Ontology Consortium Gene Ontology: tool for the unification of biology. Nature Genetics 25(1), 25–29 (2000)
6. AAT Web site, `http://www.getty.edu/research/tools/vocabularies/aat/`
7. The DBPedia Ontology, `http://wiki.dbpedia.org/Ontology`
8. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. Addison-Wesley (1999)
9. Erdmann, M., Maedche, A., Schnurr, H.-P., Staab, S.: From Manual to Semi-automatic Semantic Annotation: About Ontology-Based Text Annotation Tools. In: Proc. COLING Intl. Workshop on Semantic Annotation and Intelligent Context (2000)
10. Handschuh, S., Staab, S., Volz, R.: On deep annotation. In: Proc. Intl. World Wide Web Conference (WWW), pp. 431–438 (2003)
11. Pastorello Jr., G.Z., Daltio, J., Medeiros, C.B.: Multimedia Semantic Annotation Propagation. In: Proc. of IEEE International Symposium on Multimedia (ISM 2008), 509–514 (2008)
12. 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)