Semantic Composition of Lecture Subparts for a Personalized e-Learning

Naouel Karam, Serge Linckels, and Christoph Meinel

HPI, University of Potsdam, Germany
{naouel.karam,serge.linckels,christoph.meinel}@hpi.uni-potsdam.de

Abstract. In this paper we propose an algorithm for personalized learning based on a user's query and a repository of lecture subparts —i.e., learning objects— both are described in a subset of OWL-DL. It works in two steps. First, it retrieves lecture subparts that cover as much as possible the user's query. The solution is based on the concept covering problem for which we present a modified algorithm. Second, an appropriate sequence of lecture subparts is generated. Indeed, the different lecture subparts are only reachable when a given prerequisite is fulfilled, i.e., the learner must have a minimal background knowledge to be able to assimilate the requested learning object. Therefore, our algorithm takes into account the user's knowledge to generate a personalized lecture composition and suggests a flow of learning objects to the user.

1 Introduction

The Semantic Web aims to provide computer-processable information on the Internet. It extends the current Web through the use of standards, markup languages and related processing tools. The recent development of Semantic Web technologies has a great impact on e-Learning. Indeed, the semantic annotation of learning resources for the aim of automated retrieval and sequencing is fully in the stream of the Semantic Web. Recent efforts on standardization led to the definition of *Learning Objects* (LOs) as reusable units of educational content that can be sequenced into larger units to allow personalized learning [17,11].

The availability of LOs in electronic form increases dramatically. Hence, the task of finding the appropriate information becomes more and more awkward and time consuming.

In this paper we propose the algorithm *LectureComposer* for personalized learning that takes as input a user's query and a repository of LOs, both described in a subset of OWL-DL. The algorithm works in two steps. First, it retrieves LOs that cover as much as possible the user's query. Note that these LOs can belong to different original lectures. Second, it composes a sequence of LOs according to required prerequisites. The composition takes into account the user's knowledge about the subject in order to propose the best sequence for the retrieved LOs.

For the first issue, we propose to use the concept covering problem, recently introduced for a subset of OWL-DL [9]. It is stated as follows:

> *Given an ontology $\mathcal{T}$ and a query description Q, find a combination of concepts from $\mathcal{T}$ that contains as much as possible of common information with Q and as less as possible of extra information w.r.t. Q.*

In our solution the concept covering problem allows to select a subset of LOs from the repository that covers as much as possible the user's query.

The work presented here has been developed in the context of the Web University project [2], which aims at exploring novel internet- and IT-technologies in order to enhance university teaching and research. Our solution can be used by learners to dynamically compose a personalized lecture according to their needs and according to their current knowledge about the subject. It is able to identify the missing information in the yielded results—i.e., the data that is not available in the repository—and the required parts that are needed to fulfill the user's background knowledge. Our solution is particularly interesting for education in a self-directed learning environment, where it fosters autonomous and exploratory learning. It will be integrated to the *e-Librarian Service* "CHESt" [14] which allows students to enter questions in natural language, and yields only semantically relevant multimedia resources to the students needs.

The remainder of the paper is organized as follows. Section 2 presents an overview of Description Logics and recalls the definition of the concept covering problem. Section 3 describes the formalization of the LO composition problem. Our proposed algorithm is detailed in section 4. An illustration of the algorithm is given in section 5. Section 6 concludes the paper and outlines future work.

2   Description Logics

Description Logics (DLs) [5] are a family of knowledge representation formalisms that allow to represent the knowledge of an application domain in a structured way and to reason about this knowledge. DLs play an important role in the definition of languages for the Semantic Web [3]. Indeed, the OWL sub-language OWL-DL is based on DLs.

2.1   Preliminaries

In DLs, the conceptual knowledge of an application domain is represented in terms of *concepts* (unary predicates) such as Network and Protocol, and *roles* (binary predicates) such as hasProtocol and hasTopology. Concepts denote sets of individuals and roles denote binary relations between individuals.

Based on basic concept and role names, complex concept descriptions are built inductively using concept constructors. The different DLs languages distinguish

themselves by the kind of constructs they allow. Examples of concept constructs are the following:

- top-concept ⊤ and bottom-concept ⊥ denote all the individuals in the domain and the empty set respectively,
- conjunction ⊓, e.g., StarTopology ⊓ RingTopology denotes topologies that are both, star and ring topology (mixed topology),
- value restriction $\forall r.C$, e.g., Network ⊓∀hasTopology.StarTopology denotes networks that have only a star topology,
- existential restriction $\exists r.C$, e.g., Network ⊓ ∃hasProtocol.TCPIP denotes networks that support the TCP/IP protocol.

Concept descriptions are used to specify terminologies that define the intentional knowledge of an application domain. Terminologies are composed of *inclusion assertions* and *definitions*. The first impose necessary conditions for an individual to belong to a concept, e.g., to impose that Internet is, among other things, a network that supports only the protocol TCP/IP, one can use the inclusion assertion: Internet ⊑ Network ⊓ ∀hasProtocol.TCPIP. Definitions allow to give a meaningful name to concept descriptions, e.g., to define that a home network is a LAN based only on a star topology one can write: HomeNetwork ≐ LAN ⊓ ∀hasTopology.StarTopology.
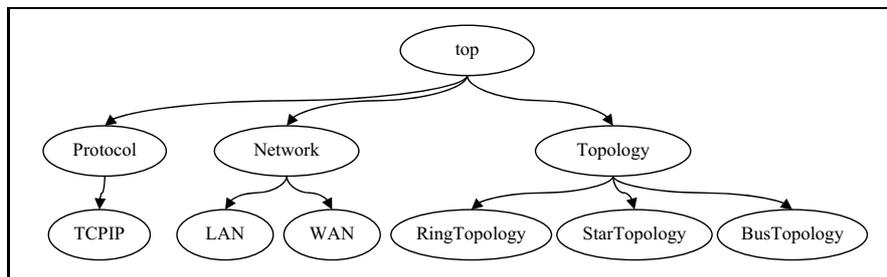
Fig. 1. Sample of a concept hierarchy about networking

DL systems provide various reasoning services. One of the most important is *subsumption*, which is the basis of the concept hierarchy (see figure 1). Formally, a concept description D subsumes a concept description C (noted $C \sqsubseteq D$) if every interpretation assigns to C a set of individuals included in the one assigned to D.

By opposition to early standard inferences in DLs, newly introduced inferences are called non standard inferences. Among these inferences we use the *least common subsumer* [4], which stands for the least concept description (w.r.t. subsumption) that subsumes a given set of concept descriptions.

Another non standard inference is the difference operation. Introduced in [16], it allows to remove from a given description all the information contained in another description. In some DLs, the difference may contain descriptions which

are not semantically equivalent. Teege defines necessary conditions for a DL to have a semantically unique difference. Those DLs are said with structural subsumption. Among others, the language $\mathcal{EL}$—which allows for conjunction ($\sqcap$), existential restriction ($\exists r.C$) and the top concept ($\top$)—satisfies this property. We use this language in the context of our project. In DLs with structural subsumption, the subsumption test can be reduced to the test of inclusion between clause[1] sets. See [16] for further details.

This definition of difference requires that the second argument subsumes the first one. However, the difference $C - D$ between two incomparable descriptions C and D can be given by constructing the least common subsumer of C and D: $C - D = C - lcs(C, D)$.

2.2   The Concept Covering Problem

The concept covering problem [9] defines a cover of a concept C w.r.t. a terminology $\mathcal{T}$ as being the conjunction of some defined concepts in $\mathcal{T}$ that share some information with Q. Based on two non standard inferences in DLs—i.e., the least common subsumer (lcs), and the difference operation—a cover is formally defined as follows:

Definition 1. *Let $\mathcal{L}$ be a DL with structural subsumption, $\mathcal{T}$ be an $\mathcal{L}$-terminology and $S_{\mathcal{T}} = \{S_i, i \in [1, n]\}$ the set of concept definitions occurring in $\mathcal{T}$. A cover of a $\mathcal{L}$-concept description $Q \not\equiv \bot$ using the terminology $\mathcal{T}$ is a conjunction E of some names S_i from $\mathcal{T}$ such that $Q - lcs(Q, E) \not\equiv Q$.*

The best cover is defined based on the remaining information in the query (called $Rest$) and in the cover (called $Miss$).

Definition 2. *Let Q be an $\mathcal{L}$-concept description and E a cover of Q using $\mathcal{T}$. The rest of Q w.r.t. to E, written $Rest_E(Q)$ is defined as follows: $Rest_E(Q) \doteq Q - lcs_{\mathcal{T}}(Q, E)$.*

The missing information of Q w.r.t. E written $Miss_E(Q)$ is defined as follows: $Miss_E(Q) \doteq E - lcs_{\mathcal{T}}(Q, E)$.

The best cover is the one with the smallest Rest and Miss.

Definition 3. *A concept description E is called a best cover of Q using a terminology $\mathcal{T}$ iff:*

- *E is a cover of Q using $\mathcal{T}$, and*
- *there does not exists a cover E' of Q using $\mathcal{T}$ such that $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$, where $Rest_E(Q) = Q - lcs_{\mathcal{T}}(Q, E)$, $Miss_E(Q) = E - lcs_{\mathcal{T}}(Q, E)$, and $<$ stands for the lexicographic order.*

An algorithm to compute the best cover based on hypergraphs theory was introduced in [9]. It is defined for DLs with structural subsumption.

---

[1] A clause is a term of conjunction that cannot be decomposed into the conjunction of other terms.

3   The LO Composition Problem

3.1   The Notions of Learning Object and Composition Flow

We suppose that lectures (or other educational content) can be split into sub-parts. Each subpart is an educational entity that is about a precise subject in the lecture. We call such a subpart a Learning Object (LO). For example, we split a 90 minutes lecture about "Internetworking" that contains 80 slides into 27 LOs.

Related projects like [7] search for a set of LOs w.r.t. a user's query that is reachable with the user's knowledge. However, if the user's knowledge is not sufficient to reach the requested LO, no result is returned at all. Our solution returns a result even if the user's knowledge is not sufficient. In that case, the missing knowledge is identified and added to the proposed composition flow.

The problem is formalized as follows. Given a learning request Q and a repository of learning objects $\{LO_1, ..., LO_n\}$, find a composition of LOs that covers the user's query as much as possible.

Some LOs may require prerequisites. This means that the user's background knowledge must satisfy the prerequisites before he can reach the requested LOs. In our solution, the LOs are organized in a way that the user can acquire such required knowledge by reading other LOs. Those complementary LOs are detected by the system and added to the resulting composition flow.

3.2   Computing the Lecture Cover

The user's query and his background knowledge are denoted Q and $\mathcal{BK}$ respectively. The knowledge offered by a learning object LO_i and the prerequisites required to reach that LO are denoted $\mathcal{LO}_i$ and $\mathcal{PR}_i$ respectively. If the user has no knowledge about the subject, then $\mathcal{BK} = \top$. In the same way, when no prerequisites are needed to reach a LO_i, then $\mathcal{PR}_i = \top$.

For the sake of clarity, we use the following notations. Given a set of LOs $S = \{LO_1, ..., LO_n\}$:

- $\mathcal{LO}_S$ denotes the knowledge offered by all the LOs in S, i.e., the conjunction $\mathcal{LO}_1 \sqcap ... \sqcap \mathcal{LO}_n$,
- $\mathcal{PR}_S$ denotes the prerequisites needed to access all the LOs in S, i.e., the conjunction $\mathcal{PR}_1 \sqcap ... \sqcap \mathcal{PR}_n$.

Self-contained Set of LOs. A self-contained set of LOs w.r.t. a background knowledge $\mathcal{BK}$ must satisfy the following conditions:

- at least one LO is reachable with the user's background knowledge $\mathcal{BK}$, and
- every remaining LO must be reachable with the background knowledge $\mathcal{BK}$ augmented by the knowledge offered by some other LOs.

Formally, a self-contained set against a background knowledge is defined as follows.

Definition 4. *Given a set of LOs $S = \{LO_1, ..., LO_n\}$, S is a self-contained set against a background knowledge $\mathcal{BK}$ if:*

- *there exists at least one $LO_i, 1 \leq i \leq n$ such that $\mathcal{BK} \sqsubseteq \mathcal{PR}_i$, and*
- *$\forall 1 \leq i \leq n$, if $\mathcal{BK} \not\sqsubseteq \mathcal{PR}_i$ then there exists a set $S_p \subseteq S$ such that $\mathcal{BK} \sqcap \mathcal{LO}_{S_p} \sqsubseteq \mathcal{PR}_i$.*

Lecture Cover. We define a lecture cover according to a user's query as a self-contained set against the user's background knowledge that shares some information with the query.

Definition 5. *Let $\mathcal{L}$ be a DL with structural subsumption, Q a $\mathcal{L}$-concept description, $S = \{LO_i, i \in [1, k]\}$ a set of LOs and $\mathcal{T} = \{\mathcal{LO}_i, i \in [1, k]\}$ a $\mathcal{L}$-terminology describing the knowledge offered by S. A lecture cover S_c is a set of LOs $S_c \subseteq S$ such that:*

- *S is a self-contained set against $\mathcal{BK}$, and*
- *$Q - lcs_{\mathcal{T}}(\mathcal{LO}_{S_c}, Q) \not\equiv Q$.*

The best lecture cover is the one with the minimal non-covered part in the query.

Definition 6. *A best lecture cover of a query Q over a set of LOs S is a set $S_c \subseteq S$ such that:*

- *S_c is a lecture cover of Q, and*
- *there exists no lecture cover S'_c of Q using S such that $\left|Q - lcs_{\mathcal{T}}(\mathcal{LO}_{S'_c}, Q)\right| < |Q - lcs_{\mathcal{T}}(\mathcal{LO}_{S_c}, Q)|$, where $<$ is the lexicographic order from the definition 3.*

3.3   Computing the Flow

Once the lecture cover S_c is computed, the identified LOs are assembled into an appropriated sequence, called the *composition flow*. The method is the following:

- at each step:
  - compute the set of LOs reachable with the user's background knowledge ($\mathcal{BK}$), noted S_r, with $S_r \subseteq S_c$,
  - add the set S_r to the composition flow,
  - remove S_r from S_c,
  - update the user's knowledge ($\mathcal{BK}'$) with the knowledge of the LOs in S_r: $\mathcal{BK}' = \mathcal{BK} \sqcap \mathcal{LO}_{S_r}$,
- the process stops when no more LO remains in the lecture cover S_c.

Remark 1. We suppose that no cycle can occur in a self-contained set S. We define the notion of cycle in a set of LOs S as follows. Let LO_1 and LO_2 be LOs in S. We say that LO_1 directly *requires* LO_2 if $\mathcal{LO}_2 \sqcap C \sqsubseteq \mathcal{PR}_1$, where C is some concept description. We call "requires" the transitive closure of the relation "directly requires". Then, S contains a cycle iff there exists a LO in S that requires itself.

4   An Algorithm for Lecture Composition

Our algorithm to solve the LO composition problem is called *LectureComposer*. It is based on the algorithm *ComputeBCov* proposed in [9] for solving the best covering problem.

4.1   The Best Covering Algorithm

The problem of computing the best covers of a concept is reduced to searching for transversals with minimal cost of a hypergraph constructed from $\mathcal{T}$ and Q. Before describing the process we recall the definition of a hypergraph and a transversal.

Definition 7. *A hypergraph $\mathcal{H}$ is a pair (Σ, Γ) of a finite set $\Sigma = \{V_1, ..., V_n\}$ and a set Γ of subsets of Σ. The elements of Σ are called vertices, and the elements of Γ are called edges.*

A transversal of a hypergraph is a subset that hits all the edges of the hypergraph. Formally, it is defined as follows.

Definition 8. *A set $T \subset \Sigma$ is a transversal of $\mathcal{H}$ if for each $\epsilon \in \Gamma$, $T \cap \epsilon \neq \emptyset$. A transversal is minimal if no proper subset T' of T is a transversal.*

According to [9], we build the hypergraph $\mathcal{H}_{SQ}$ corresponding to a concept Q and a set of LOs $S = \{LO_i, i \in [1, k]\}$ as follows:

- each concept name $\mathcal{LO}_i$, $i \in [1, k]$ is associated with a vertex $V_{\mathcal{LO}_i}$, and
- each clause $A_i \in Q$ is associated to an edge e_{A_i} with $e_{A_i} = \{V_{\mathcal{LO}_i} \mid A_i \in_{\equiv} lcs(Q, \mathcal{LO}_i)\}$, where $\in_{\equiv}$ stands for membership modulo equivalence[2] of clauses.

The algorithm follows a classical approach for computing the minimal transversals with some improvements. It works in n steps, where n is the number of edges in the hypergraph. Starting from an empty set of transversals, it explores each edge of the hypergraph—one edge in each step—and generates a set of candidate transversals by computing all the possible unions between the candidates generated in the previous step and each vertex in the considered edge. At each step, the non-minimal candidate transversals are pruned. Heuristics and combinatorial optimization techniques [6] are used to discard non-minimal transversals at early steps.

As our definition of the best cover does not take into account a preference criteria between best covers, it is not necessary to compute all the transversals of the hypergraph. Thus, we adopt a deep first strategy for computing transversals as described in [13]. The idea is to compute a transversal of the partial hypergraph and then add the next hyperedge to it. Instead of computing all the transversals that follow, pick only one and add the next hyperedge. When no

---

[2] Two concept descriptions C and D are called *equivalent* ($C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.

more hyperedge remains, the algorithm backtracks to the previous level and picks the next transversal, etc. In this way the first transversal is displayed quickly. As we need only one transversal, the process can stop after the computation of the first transversal. We call the modified algorithm *ComputeFirstBCov*.

In summary, our algorithm computes from all possible best concept covers only the first one that it finds. The advantage is that a solution—i.e., a best cover—is found quickly, because not all combinations in the hypergraph have to be explored.

4.2   The Composition Algorithm

Our algorithm (depicted in figure 2) takes as input a query Q and a set of LOs S. First, the best cover of Q w.r.t. S is computed (line 1). As explained in section 4.1, this corresponds to identifying the LOs that cover best the user's query. The result of the cover is the first transversal Tr of the corresponding hypergraph $\mathcal{H}_{SQ}$.

Require: a query Q, a set of LOs $S = \{LO_i, i \in [1, k]\}$.
Ensure: Tr
 1: $Tr = ComputeFirstBCov(\mathcal{S}, Q)$
 2: $P = \mathcal{PR}_{Tr} - lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr})$
 3: **while** $P \not\equiv \top$ **do**
 4:     **for** each edge $E \in \varGamma_P$ **do**
 5:         $Tr \leftarrow$ the next transversal obtained by adding E to $\varGamma$.
 6:     **end for**
 7:     $P = \mathcal{PR}_{Tr} - lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr})$
 8: **end while**

Fig. 2. The algorithm LectureComposer

Second, the prerequisites—to reach the LOs in the lecture cover—are updated (line 2). All the prerequisites that are not covered by the user's background knowledge or by the knowledge given by other LOs in Tr are identified.

Third, the best cover for the required prerequisites is computed. Technically, the hyperedges of each clause in the prerequisites' definitions are added incrementally following the deep-first strategy, and the transversal of the obtained hypergraph is computed (lines $4-6$). The notation $\varGamma_P$ is used to denote the set of hyperedges corresponding to the clauses of the concept description P.

Again the prerequisites are updated as described above (line 7).

The process is repeated until no more uncovered prerequisites remain (line 3).

We ensure that the algorithm terminates because all the prerequisites are in the repository and because no cycle can occur (see remark 1).

The next theorem proves that the algorithm LectureComposer returns a transversal Tr that corresponds to the best composed lecture cover of the query Q over a set of LOs S.

Theorem 1. *Let $\mathcal{L}$ be a DL with structural subsumption, Q a $\mathcal{L}$-concept description, $S = \{LO_i, i \in [1,k]\}$ a set of LOs. Then, $Tr = \textsf{LectureComposer}(Q,S)$ is the best composed lecture cover of Q over S.*

Proof. According to definitions 4, 5 and 6 about the best lecture cover of Q over S, Tr must satisfy the following properties:

(1) there exists at least one $LO_i, 1 \leq i \leq n$ such that $\mathcal{BK} \sqsubseteq \mathcal{PR}_i$,
(2) $\forall 1 \leq i \leq n$, if $\mathcal{BK} \not\sqsubseteq \mathcal{PR}_i$ then there exists a set $Tr \subseteq S$ such that $\mathcal{BK} \sqcap \mathcal{LO}_{Tr} \sqsubseteq \mathcal{PR}_i$,
(3) $Q - lcs_{\mathcal{T}}(\mathcal{LO}_{Tr}, Q) \not\equiv Q$, where $\mathcal{T} = \{\mathcal{LO}_i, i \in [1,k]\}$ is the $\mathcal{L}$-terminology describing the knowledge offered by S,
(4) there exists no lecture cover Tr' of Q using S such that $|Q - lcs_{\mathcal{T}}(\mathcal{LO}_{Tr'}, Q)| < |Q - lcs_{\mathcal{T}}(\mathcal{LO}_{Tr}, Q)|$, where $<$ is the lexicographic order of definition 3.

Points (3) and (4) follow directly from the definition of the best cover of a concept w.r.t. a terminology.

Let us prove point (2). The algorithm returns a set Tr verifying:

$$\mathcal{PR}_{Tr} - lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr}) = \top$$

This is equivalent to: $\prod_{1 \leq i \leq n} Pr_i - lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr}) = \top$, which again is equivalent to: $Pr_i - lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr}) = \top, \forall 1 \leq i \leq n$. From the definition of the difference operator we now that, if $C - D = E$ then $C \equiv E \sqcap D$, so we can write:

$$lcs_{\mathcal{T}}(\mathcal{BK} \sqcap \mathcal{LO}_{Tr}, \mathcal{PR}_{Tr}) \equiv \mathcal{PR}_i, \forall 1 \leq i \leq n$$

It follows that:
$$\mathcal{BK} \sqcap \mathcal{LO}_{Tr} \sqsubseteq \mathcal{PR}_i, \forall 1 \leq i \leq n$$

Thus, we have proven point (2). Let us now turn to point (1). We have proven in point (2) that $\mathcal{LO}_{Tr} \sqcap \mathcal{BK} \sqsubseteq \mathcal{PR}_i, \forall 1 \leq i \leq n$. Now we must prove that for at least one $i \in [0,1]$ we have $\mathcal{BK} \sqsubseteq \mathcal{PR}_i$. In other terms, this means that for at least one $i \in [0,1]$ we must prove that $\mathcal{LO}_{Tr} \not\sqsubseteq \mathcal{PR}_i$.

Let $j \in [0,1]$, we have: $\mathcal{BK} \sqcap \mathcal{LO}_{Tr} \sqsubseteq \mathcal{PR}_j$. For all $LO_i \in \mathcal{LO}_{Tr}, \mathcal{LO}_i \not\sqsubseteq \mathcal{PR}_j$, otherwise a cycle occurs in Tr. But as stated in remark 1, we suppose that S is acyclic. Thus, there exists at least one j such that $\mathcal{BK} \sqsubseteq \mathcal{PR}_j$ and we have proven point (1).

5   Illustrating Example

The example is based on a lecture about networking that can be found in the online tele-TASK archive [1]. A sample of the different LOs in the lecture with their corresponding prerequisites is shown in figure 3.

The example shows that the LO introducing computer networks ($\mathcal{LO}_1$) can be accessed by beginners because it requires no initial knowledge. The LO about

$$\begin{array}{ll}
\mathcal{LO}_1 \doteq \exists \text{communicationSystem.Network} & \mathcal{PR}_1 \doteq \top \\
\mathcal{LO}_2 \doteq \exists \text{network.WAN} & \mathcal{PR}_2 \doteq \exists \text{communicationSystem.Network} \\
\mathcal{LO}_3 \doteq \exists \text{network.LAN} & \mathcal{PR}_3 \doteq \exists \text{communicationSystem.Network} \\
\mathcal{LO}_4 \doteq \exists \text{communication.Protocol} & \mathcal{PR}_4 \doteq \exists \text{communicationSystem.Network} \\
\mathcal{LO}_5 \doteq \exists \text{protocol.TCPIP} & \mathcal{PR}_5 \doteq \exists \text{communication.Protocol} \\
\mathcal{LO}_6 \doteq \exists \text{structure.Topology} & \mathcal{PR}_6 \doteq \exists \text{communicationSystem.Network} \\
\mathcal{LO}_7 \doteq \exists \text{topology.StarTopology} & \mathcal{PR}_7 \doteq \exists \text{structure.Topology} \sqcap \exists \text{network.LAN} \\
\mathcal{LO}_8 \doteq \exists \text{topology.RingTopology} & \mathcal{PR}_8 \doteq \exists \text{structure.Topology} \\
\mathcal{LO}_9 \doteq \exists \text{topology.BusTopology} & \mathcal{PR}_9 \doteq \exists \text{structure.Topology}
\end{array}$$

**Fig. 3.** Examples of LO descriptions and their corresponding prerequisites

star topology ($\mathcal{LO}_7$) requires some knowledge about topologies in general ($\mathcal{LO}_6$) and about LAN ($\mathcal{LO}_3$). Therefore, this LO can only be accessed if the user's background knowledge $\mathcal{BK}$ fulfills this requirements.

Let us suppose that the user wants to learn something about topologies in general and star topology in particular, and that he has a basic knowledge about computer networks. Formally, this is written as follows:

$$Q \doteq \exists \text{structure.Topology} \sqcap \exists \text{topology.StarTopology}$$
$$\mathcal{BK} \doteq \exists \text{communicationSystem.Network}$$

The associated hypergraph $\mathcal{H}_{SQ} = (\Sigma, \Gamma)$ consists on the vertices $\Sigma = \{V_{\mathcal{LO}_1}, ..., V_{\mathcal{LO}_9}\}$ and the edges $\Gamma = \{e_{\exists \text{structure.Topology}}, e_{\exists \text{topology.StarTopology}}\}$. The only minimal transversal is $Tr = \{V_{\mathcal{LO}_6}, V_{\mathcal{LO}_7}\}$ (see figure 4). It covers completely the query.
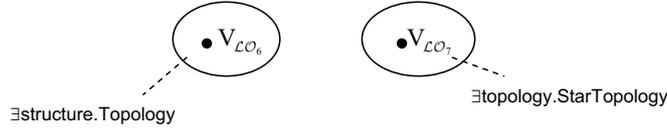
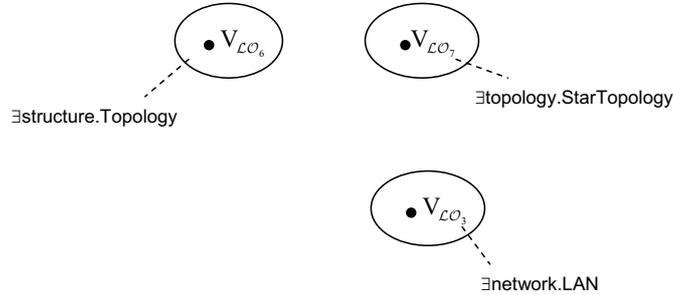**Fig. 4.** The hypergraph $\mathcal{H}_{SQ}$

The prerequisites required for the set Tr are:

$$\mathcal{PR}_{Tr} = \exists \text{communicationSystem.Network} \sqcap \exists \text{structure.Topology} \sqcap \exists \text{Network.LAN}$$

The part of prerequisites that are not covered by the user's background knowledge or by other LOs is:

$$P = \exists \text{network.LAN}$$

The new hypergraph $\mathcal{H}'_{SQ}$ that is obtained by adding the hyperedges in Γ_P is shown in figure 5.

Fig. 5. The hypergraph $\mathcal{H}'_{SQ}$

The new minimal transversal is then $Tr = \{V_{\mathcal{L}\mathcal{O}_6}, V_{\mathcal{L}\mathcal{O}_7}, V_{\mathcal{L}\mathcal{O}_3}\}$. The prerequisites needed for it are:

$$\mathcal{PR}_{Tr} = \exists \text{communicationSystem.Network} \sqcap \exists \text{structure.Topology} \sqcap \exists \text{network.LAN}$$

The part of prerequisites not covered by the user background knowledge or by other LOs is:

$$P = \top$$

Thus, the algorithm stops and returns LO_6, LO_7 and LO_3. Although, LO_3 is not directly requested by the user—i.e., it is not a cover of the user's query— it is needed to fulfill the missing prerequisite since the user must have some knowledge about LAN before being able to learn something about star topology.

6   Discussion

In this paper we have proposed a novel algorithm for personalized lecture composition based on lecture subparts. We used two non-standard inferences in DLs— i.e., the least common subsumer (lcs), and the difference operation—to compute the best cover of the user's query w.r.t. a repository of LOs. Then, the LOs are assembled into a composition flow. The algorithm takes into account the user's knowledge as well as the sequencing constraints between the LOs in order to retrieve a comprehensive and self-contained composition flow.

A lot of recent work has focused on personalized e-learning and composition of LOs [12,7,8,10]. The work presented in [7] is closely related to ours. The authors introduced a new definition of the concept covering problem that eliminates the limitation of the DLs to have structural subsumption. It is based on the concept abduction problem (CAP) which was introduced in [15] to provide an explanation when subsumption does not hold. But, the proposed algorithm returns a cover which is not necessarily the best one.

As DLs with structural subsumption are expressive enough for our application, we chose to use the algorithm of Hacid & al. [9] because it always returns the best cover. Also compared to [7], the novelty of our approach is that it always

proposes a solution to the user. When the user's knowledge is not sufficient, our algorithm looks for complementary LOs and adds them to the composition flow in order to fill the missing knowledge.

Currently, we are improving our prototype that will be integrated in the Web interface of the online tele-TASK archive [1]. Experiments on real scenarios will be conducted on a set of lectures about "Internetworking". For this purpose, an appropriated domain ontology is under development.

In future work, we will improve our algorithm by taking into account further criteria for the best composition. We will consider minimizing a number of parameters like the number of retrieved LOs, optimizing the length of the composition flow, and the number of original lectures involved in the composition flow.

References

1. Tele-TASK – Teleteaching Anywhere Solution Kit. http://www.tele-task.de.
2. Web University project. http://www.hpi.uni-potsdam.de/∼meinel/research/web_ university.html.
3. F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In *F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, Festschrift in honor of J org Siekmann, Lecture Notes in Artificial Intelligence.*, 2003.
4. F. Baader, R. Küsters, and R. Molitor. Computing Least Common Subsumers in Description Logics with Existential Restrictions. In T.Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101. Morgan Kaufmann, 1999.
5. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge: University Press, 2003.
6. Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, and Farouk Toumani. On automating web services discovery. *The VLDB Journal*, 14(1):84–96, 2005.
7. Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Azzurra Ragone. Semantic-based automated composition of distributed learning objects for personalized e-learning. In *The Semantic Web: Research and Applications 2nd European Semantic Web Conference ESWC 05*, volume 3532, pages 633–648. Springer-Verlag, 2005.
8. Robert G. Farrell, Soyini D. Liburd, and John C. Thomas. Dynamic assembly of learning objects. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 162–169, New York, NY, USA, 2004. ACM Press.
9. M. Hacid, A. Leger, C. Rey, and F. Toumani. Computing concept covers: A preliminary report. In *Workshop on Description Logics*, 2002.
10. N. Henze. Personal readers: Personalized learning object readers for the semantic web, 2005.
11. A. Ip, A. Young, and I. Morrison. Learning objects - whose are they? In *Proceedings of the 15th Annual Conference of the National Advisory Committee on Computing Qualifications ISBN 0-473-08747-2*, pages 315–320, 2002.

12. Jelena Jovanovic, Dragan Gasevic, and Vladan Devedzic. Dynamic assembly of personalized learning content on the semantic web. In York Sure and John Domingue, editors, *ESWC*, volume 4011 of *Lecture Notes in Computer Science*, pages 545–559. Springer, 2006.
13. Dimitris J. Kavvadias and Elias C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *J. Graph Algorithms Appl.*, 9(2):239–264, 2005.
14. Serge Linckels, Stephan Repp, Naouel Karam, and Christoph Meinel. The virtual tele-task professor—semantic search in recorded lectures. In Ingrid Russell, Susan Haller, J.D. Dougherty, Susan Rodger, and Gary Lewandowski, editors, *ACM SIGCSE'07 Technical Symposium on Computer Science Education*, pages 50–54, New York, NY, USA, 2007. ACM Press.
15. Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Abductive matchmaking using description logics. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 337–342. Morgan Kaufmann, 2003.
16. G. Teege. Making the Difference: A Subtraction Operation for Description Logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR'94: Principles of Knowledge Representation and Reasoning*, pages 540–550. Morgan Kaufmann, San Francisco, California, 1994.
17. Gottfried Vossen and Peter Jaeschke. Learning objects as a uniform foundation for e- learning platforms. *IDEAS*, 2003.