ON THE PROBLEMS OF REPRESENTING LEAST COMMON SUBSUMER AND COMPUTING APPROXIMATION IN DLS

Chan Le Duc and **Nhan Le Thanh**
Laboratoire I3S, Université de Nice - Sophia Antipolis, France
Email: cleduc@i3s.unice.fr, Nhan.Le-Thanh@unice.fr

Abstract. Design and translation between of DL-based ontologies require inference services as Least Common Subsumer and Approximation. However, in general the size of concept descriptions obtained from these inferences remains exponential in the size of input. In this paper, we propose a compact representation for description trees of Least Common Subsumer. This representation is based on a specific data structure aimed at avoiding repetition of parts in trees. Second, we will show that at the worst case the size of approximation in $\mathcal{ALE}$ of an $\mathcal{ALC}$-concept description can be doubly exponential in the size of input in spite of the proposed compact representation for Least Common Subsumer.

1. Introduction

Description Logics can be used as a formalism for design of ontologies in an application domain. In order to reduce the size of ontologies, different ontologies for different subdomains or user profiles are derived from a shared common ontology. These derived ontologies share atomic concepts, atomic roles and defined basic concepts from the common ontology. In some cases, there is a difference in the expressiveness of Description Logics used for derived ontologies. Thus arises the problem of translating a concept from a derived ontology into another. This problem is known as approximation from L_1-Description Logic to L_2-Description Logic. An algorithm for computing the approximation where L_1=$\mathcal{ALC}$ and L_2=$\mathcal{ALE}$, which is presented in [1], returns a concept description whose size is doubly exponential in the size of input. This algorithm uses another exponential algorithm which allows us to compute the Least Common Subsumer (*lcs*) of two concept descriptions in $\mathcal{ALE}$. As shown in [4], the exponential size of *lcs* cannot be avoided if we use the ordinary representation for expressing concept descriptions. In fact, the size of concept descriptions grows exponentially because of normalization prior to computing *lcs*. If there is a compact representation for concept descriptions in which the exponential blow-up is buried (but not eliminated), we can improve the *lcs* computing by an algorithm requiring polynomial space. This fact does not bring about a change in the computing complexity for *lcs* since PSPACE = NSPACE and NP $\subseteq$ PSPACE. However, we may find an application of this improvement in computing which does not require the ordinary representation of concept descriptions.

In this paper, we will present a compact representation for *lcs*. This representation results from a normalization method for $\mathcal{ALE}$-concept descriptions in which the interaction between value and existential restrictions are compacted. The idea supporting this compactness is to introduce ϵ-edges into $\mathcal{ALE}$-description trees aimed at avoiding the repetition of parts of the tree. This repetition comes from applications of the propagation rules. We will show that the compact representation for *lcs* can be obtained from computing the product of ϵ-description trees. An algorithm for computing *lcs* requiring polynomial space results naturally from this. Second, we will introduce an example that shows that an exponential algorithm for computing the approximation in $\mathcal{ALE}$ of an $\mathcal{ALC}$-concept description does not exist in spite of the compact representation for *lcs*.

2. Preliminaries

Let N_C be a set of primitive concepts and N_R be a set of primitive roles. Language $\mathcal{ALE}$ uses the following constructors to build concept descriptions : conjunction ($C \sqcap D$), value restriction ($\forall r.C$), existential restriction ($\exists r.C$), primitive negation ($\neg P$), concepts TOP and BOTTOM. Language $\mathcal{ALC}$ uses all of these constructors together with disjunction ($C \sqcup D$).

Let Δ be a non-empty set of individuals. Let $\cdot^{\mathcal{I}}$ be a function that transforms each primitive concept $P \in N_C$ into $P^I \subseteq \Delta$ and each primitive role $r \in N_R$ into $r^I \subseteq \Delta\times \Delta$. The semantics of a concept description are inductively defined owing to the interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ in the table below.

Syntax	Semantics	Syntax	Semantics
$\top$	Δ	$\bot$	$\emptyset$
$C \sqcap D$	$C^I \cap D^I$	$C \sqcup D$	$C^I \cup D^I$
$\forall r.C$, $r \in N_R$	$\{x \in \Delta \vert \forall y{:}(x,y) \in r^I \rightarrow y \in C^I\}$	$\neg C$	$\Delta \setminus C^I$
$\exists r.C$, $r \in N_R$	$\{x \in \Delta \vert \exists y{:}(x,y) \in r^I \wedge y \in C^I\}$		

Subsumption Let C, D be concept descriptions. D subsumes C, $C \sqsubseteq D$, iff $C^I \subseteq D^I$ for all interpretations $\mathcal{I}$.
Least Common Subsumer Let C_1, C_2 be concept descriptions in a DL language. The least common subsumer C is a least common subsumer of C_1, C_2 (for short $lcs(C_1, C_2)$) iff $C_i \sqsubseteq C$ for all i , and if C' is a concept description such that $C_i \sqsubseteq C'$ for all i, then $C \sqsubseteq C'$.
Approximation Let C be a concept descriptions in a L_1 and D be a concept descriptions in a L_2 where L_1, L_2 are DL languages. D is called upper L_2-approximation of C ($D = approx_{\mathcal{L}_1}(C)$ for short) iff $C \sqsubseteq D$ and, if $C \sqsubseteq D'$ and $D' \sqsubseteq D$, then $D' \equiv D$ for all L_2-concept description D'.

Definition 1. [3]. An $\mathcal{ALE}$-concept description is called *propagated normal* if the following rules are exhaustively applied : (1) $\forall r.E \sqcap \forall r.F \rightarrow \forall r.(E \sqcap F)$; (2) $\forall r.E \sqcap \exists r.F \rightarrow \forall r.\ E \sqcap \exists r.(E \sqcap F)$; (3) $\forall r.\top \rightarrow \top$; (4) $E \sqcap \top \rightarrow E$; (5) $P \sqcap \neg P \rightarrow \bot$ for each $P \in N_C$; (6) $\exists r.\bot \rightarrow \bot$; (7) $E \sqcap \bot \rightarrow \bot$, where E, F are two $\mathcal{ALE}$-concept descriptions and $r \in N_R$.

Description tree of an $\mathcal{ALE}$-concept description [3] Description tree of an $\mathcal{ALE}$-concept description is a tree in the form of $\mathcal{G} = (N, E, n_0, l)$ where N is a finite set of nodes of $\mathcal{G}$; E is a finite set of edges labeled with role names r ($\exists$-edges) or with $\forall r$ ($\forall$-edges); n_0 is the root of $\mathcal{G}$; l is a labeling function mapping the nodes in N to finite set $\{P_1, ..., P_k\}$ where $P_i \in N_C$, $P_i = \neg P$ for some $P \in N_C$, or $P_i = \bot$. The empty label corresponds to the concept TOP.

3. ϵ-description tree for $\mathcal{ALE}$-concept description

As mentioned in [3], the application of normalization rule 2 (as specified in Definition 1) to an $\mathcal{ALE}$-concept description can increase the latter's size exponentially. In this section, we propose a specific data structure for the $\mathcal{ALE}$-description trees by using empty edges, so-called ϵ-edges. Hence, an application of normalization rule 2 can be interpreted as an addition of ϵ-edge. We suppose that all $\mathcal{ALE}$-concept descriptions considered in this section are normalized by all normalization rules in Definition 1 with the exception of rule 2. Furthermore, for the sake of simplicity, we can assume that $N_R = \{r\}$. All obtained results can be applied to a general set N_R. We denote $\mathcal{G}^\epsilon = (V_G, E_G \cup E^\epsilon_G, v_G, l_G)$ where E^ϵ_G is the set of ϵ-edges and $v^k \in V_G$ as a node at the level k of the tree.

Definition 2. (neighbourhood) Let C be an $\mathcal{ALE}$-concept description and $\mathcal{G} = (V_G, E_G, v^0_G, l_G)$ be its description tree. At level 0 of $\mathcal{G}$, there is a unique neighbourhood, we denote $N^0 = \{v^0_G\}$. For each level $k > 0$ of $\mathcal{G}$, the set of neighbourhoods at level k, N^k is inductively defined as follows $1 \leq k \leq |G|$ ($|G|$ is the height of $\mathcal{G}$) : for each $n^{k-1} \in N^{k-1}$ where $n^{k-1} = \{v^{k-1}_1, ..., v^{k-1}_m\}$, the corresponding elements of N^k are built as follows : $\{v^k_{i1}, ..., v^k_{in}\}$

$\in N^k$ where $v_{i1}^{k-1}\forall rv^k, ..., v_{in}^{k-1}\forall rv_{in}^k$ for $1 \leqslant i1,...,in \leqslant m$ and $\{\{v_{i1}^k, ..., v_{in}^k\} \cup \{v_j^k\}\} \in N^k$ for all $v_j^{k-1} \in n^{k-1}$ where $v_i^{k-1} r v_j^k$.

K-neighbourhoods correspond to the applications of rules 2 and 1 to description tree at level k. In order to avoid copying subtrees in the normalization, we use relevant ϵ-edges to link nodes of a k-neighbourhood. The following definition proposes such a way to build description tree with ϵ-edges (More precisely, the tree now becomes a graph !).

Definition 3. (ϵ-description tree) Let C be an $\mathcal{ALE}$-concept description and $\mathcal{G}=(V_G, E_G, v_G^0, l_G)$ its description tree. For each level $k \geq 0$ of $\mathcal{G}$, for each $n^k \in N^k$ and for each couple of nodes $v_i^k, v_j^k \in n^k$: (1) If $v_i^{k-1}\forall rv_i^k, v_j^{k-1}\forall rv_j^k$ and $v_i^{k-1}\epsilon v_j^{k-1}$, ϵ-edge $v_i^k \epsilon v_j^k$ is added. (2) If $v_i^{k-1} r v_i^k, v_j^{k-1}\forall r v_j^k$, ϵ-edge $v_i^k \epsilon v_j^k$ is added.

The ϵ-concept description construction as in Definition 3 can take at most an exponential time in the size of C. An equivalent definition - however taking a polynomial time - for ϵ-description tree is described in [9]. We need the following algorithm in order to transform an ϵ-description tree into usual description tree corresponding to the concept description normalized by the normalization rules. In addition, this algorithm works also for extended ϵ-description that will be presented in the next section. Note that for ϵ-description tree (not extended), the flag of each node $f(v)$ is the predecessor node of v.

Algorithm 1. ***(operator B)***
Input: (Extended) ϵ-description tree $\mathcal{G}^\epsilon = (V_G, E_G \cup E_G^\epsilon, v_0, l_G)$.
Output: $\mathcal{ALE}$-description tree $\mathcal{G}$.

(1) *At level 0, an 0-neighbourhood $[v_0, v_{0,1}, ..., v_{0,m}]$, which is the root of $\mathcal{G}$, is created where the nodes $v_0, v_{0,1}, ..., v_{0,m}$ are connected by ϵ-edges. The label of 0-neighbourhood is evaluated by an application of function* val *to labels of nodes in the 0-neighbourhood.*
(2) *For each* (k-1)*-neighbourhood $(v_1^{k-1}, ..., v_m^{k-1})$ of $\mathcal{G}^\epsilon$ corresponding to (k-1)-node of $\mathcal{G}$, k-nodes and edges of $\mathcal{G}$ are created from* k*-neighbourhoods of $\mathcal{G}^\epsilon$ as follows:*
   (a) *for all $\forall r$-successors of nodes $v_1^{k-1}, ..., v_m^{k-1}$, a* k*-neighbourhood is determined. An $\forall r$-edge is added to connect the* (k-1)*-neighbourhood with the* k*-neighbourhood. The label of the* k*-neighbourhood is evaluated by an application of function* val *to labels of nodes in the* k*-neighbourhood.*
   (b) *for each $\exists r$-successor of nodes $v_1^{k-1}, ..., v_m^{k-1}$ and destinations of the ϵ-edges exiting from this $\exists r$-successor where the flags of these destinations must belong to $\{v_1^{k-1}, ..., v_m^{k-1}\}$, a* k*-neighbourhood is determined . An $\exists r$-edge is added to connect the* (k-1)*-neighbourhood with the* k*-neighbourhood. The label of the* k*-neighbourhood obtained is evaluated by an application of function* val *to labels of nodes in the* k*-neighbourhood.*
(3) *Apply normalization rules (4), (6) and (7) for the tree $\mathcal{G}$ obtained.*

Note. The function *eval* performs computations on label expressions of nodes which are composed of intersections, unions of concept name sets. It returns a union of the label expressions of nodes if these nodes are connected by ϵ-edges. Otherwise, it returns an intersection of the label expressions of nodes. In addition, it treats also negative concept names and concepts TOP, BOTTOM.

Algorithm 1 takes an exponential time and requires polynomial space. In fact, it requires at most a memory to keep a series of neighbourhoods n_1^1,...,n_1^k where k is the depth of the tree. This series corresponds to a path from the root to a leaf of the resulting tree. Moreover, together with this path, the neighbourhoods of each node along the path also need to be kept; the number of neighbourhoods of each node is polynomial in k. Thus, *Algorithm* 1 is in PSPACE in k. Next, Proposition 1 establishes the equivalence between the tree obtained from the normalization (as specified in Definition 1) and the tree obtained from *Algorithm 1.*

Proposition 1. *Let C be an $\mathcal{ALE}$-concept description and $\mathcal{G}^\epsilon$ be its ϵ-description tree. The tree **B**($\mathcal{G}^\epsilon$) is identical to the description tree $\mathcal{G}$ corresponding to the normalized concept description of C.*

4. Product of ϵ-description trees

4.1. **Definition and Description.** According to the previous section, the semantics of an ϵ-description tree do not change if we add to it ϵ-edges, namely ϵ-cycle, these two extremities of which coincide. This addition of ϵ-cycles allows every ϵ-edge of the ϵ-description trees to reappear in the product. Furthermore, each node v of an ϵ-description tree possesses a flag $f(v)$ that is set by default to the node $v'$where $(v'ev)$, $e \neq \epsilon$. The flag of each node and the entrance ϵ-edges of the node allow the operator $\mathbf{B}$ to determine which neighbourhood this node belongs to. Formally, except for the presence of ϵ-edges and flags, the definition of product of two ϵ-description trees is not different from the product definition of two $\mathcal{ALE}$-description trees given in [3].

Definition 4. Let C, D be $\mathcal{ALE}$-concept descriptions and $\mathcal{G}^\epsilon = (V_G, E_G \cup E_G^\epsilon, v_0, l_G)$, $\mathcal{H}^\epsilon = (V_H, E_H \cup E_H^\epsilon, v_0, l_H)$ be ϵ-description trees with ϵ-cycles, respectively, of C, D. If $l_G = \{\bot\}$ ($l_H = \{\bot\}$) then we define $\mathcal{G}^\epsilon \times \mathcal{H}^\epsilon$ by replacing each node w (v) in $\mathcal{H}^\epsilon$ ($\mathcal{G}^\epsilon$) by (v_0, w) ((v, w_0)). Otherwise, the product is defined by induction on the depth of the trees. The node (v_0, w_0) labelled with $l_G(v_0) \cap l_H(w_0)$ is the root of $\mathcal{G}^\epsilon \times \mathcal{H}^\epsilon$.

- For each $\forall r$-successor (r-successor) v of v_0 in $\mathcal{G}^\epsilon$ and each $\forall r$-successor (r-successor) w of w_0 in $\mathcal{H}^\epsilon$, we obtain a $\forall r$-successor (v, w) ($\exists r$-successor (v, w)) of (v_0, w_0) in $\mathcal{G}^\epsilon \times \mathcal{H}^\epsilon$ that is the root of the inductively defined product of $\mathcal{G}^\epsilon(v)$ and $\mathcal{H}^\epsilon(w)$. The flag of (v, w), $f(v, w)$, is set at $(f(v), f(w))$.
- For each ϵ-successor v of v_0 in $\mathcal{G}^\epsilon$ and w of w_0 in $\mathcal{H}^\epsilon$, we obtain an ϵ-successor (v, w) of (v_0, w_0) in $\mathcal{G}^\epsilon \times \mathcal{H}^\epsilon$ that is the root of the inductively defined product of $\mathcal{G}^\epsilon(v)$ and $\mathcal{H}^\epsilon(w)$. The flag of (v, w), $f(v, w)$, is set at $(f(v), f(w))$.
- Otherwise, for each e-successor v of v_0 in $\mathcal{G}$ and each f-successor w of w_0 in $\mathcal{H}$ where $e, f \neq \epsilon, e \neq f$, we obtain a node (v, w) that is the root of the inductively defined product of $\mathcal{G}^\epsilon(v)$ and $\mathcal{H}^\epsilon(w)$. The flag of (v, w), $f(v, w)$, is set at $(f(v), f(w))$.

The last case in the Definition 4 is important since there may exist an ϵ-edge of the product, which results from an ϵ-edge and ϵ-cycle, enters into a node created in this case.

Example 1. Let $D_1 := \exists r.(A \sqcap \forall r.C) \sqcap \exists r.(B \sqcap \forall r.B) \sqcap \forall r.(C \sqcap \exists r.A)$ and $D_2 := \exists r.(A \sqcap B \sqcap C \sqcap \exists r.(A \sqcap B \sqcap C))$ where $A, B, C \in N_C$, $r \in N_R$.

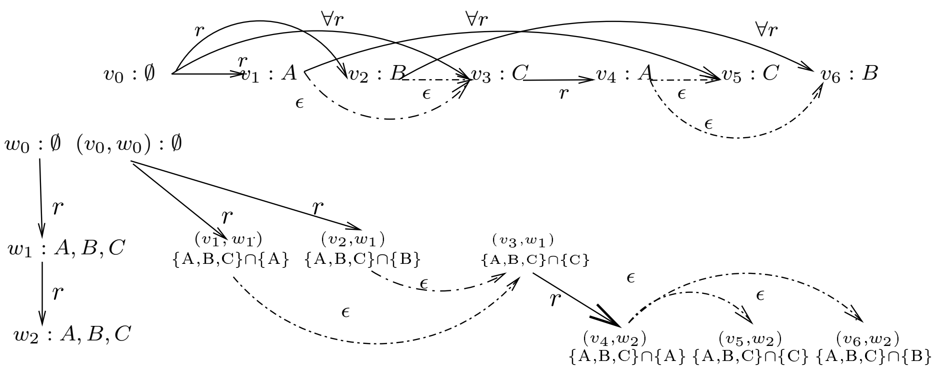

Figure 1. Product

The ϵ-description tree of the product of $\mathcal{G}_{D_1}^\epsilon$, $\mathcal{G}_{D_2}^\epsilon$ is described in the Figure 1. Implicitly, there is an ϵ-cycle on each node of $\mathcal{G}_{D_1}^\epsilon$ and $\mathcal{G}_{D_2}^\epsilon$. The label expressions of the product will be evaluated whenever it is necessary.

A tree built from *Definition* 4 can contain nodes the entrance edges of which are only ϵ-edges. An ϵ-description tree which contains such nodes is called *extended* ϵ-description tree. A neighbourhood which includes such a node is called *extended* neighbourhood. The following theorem affirms the distributive property of operator $\mathbf{B}$ over the product computed as in Definition 4.

Theorem 1. *Let $\mathcal{G}^\epsilon$, $\mathcal{H}^\epsilon$ be ϵ-description trees. We have $\boldsymbol{B}(\mathcal{G}^\epsilon \times \mathcal{H}^\epsilon) \equiv \boldsymbol{B}(\mathcal{G}^\epsilon) \times \boldsymbol{B}(\mathcal{H}^\epsilon)$.*

A proof of the Theorem can be found in [9]. *Proposition 1* and *Theorem* 1 allow us to represent *lcs* of two $\mathcal{ALE}$-concept descriptions C, D as an extended ϵ-description tree whose size is product of the sizes of the ϵ-description trees corresponding to C and D. To obtain the ordinary representation of *lcs*, one needs an application of operator $\mathbf{B}$ for this extended ϵ-description tree.

5. On the computation of Approximation $\mathcal{ALC}$-$\mathcal{ALE}$

In [1], a doubly exponential algorithm is proposed for approximation $\mathcal{ALC}$-$\mathcal{ALE}$. In this algorithm, the approximation is computed via lcs. A question left open by the authors concerns the existence of an *exponential algorithm* for computing the approximation. In the first attempt at finding an answer to this question, we hoped that if there is a method for obtaining a polynomial representation for lcs, a such method may be applied for reducing the exponential blow-up caused by the distribution of disjunctions over conjunctions in the normalization for $\mathcal{ALC}$-concept descriptions. However, though the polynomial representation for *lcs* presented in section 4 allows us to reduce the size of approximation, this reduction does not change the complexity class. In this section, we will construct an $\mathcal{ALC}$-concept description C such that the size of $approx_{ALE}(C)$ is doubly exponential in the size of C. Therefore, the doubly exponential size of approximation cannot be avoided.

We need some notations for using the approximation algorithm in [1]. Let C is an $\mathcal{ALC}$-concept description. $PRIM(C)$ denotes a set of all (negated) concept names occurring on the top-level conjunction of C. $VAL(C)$ is conjunction of all C' occurring in value restrictions of form $\forall r.C'$ on top-level of C. If there is no value restriction on top-level of C then $VAL(C)$ $= \top$. $EX(C)$ is a set of all C' occurring in existential restrictions of form $\exists r.C'$on top-level of C. Normal form of C : $C = C_1 \sqcup ... \sqcup C_n$ where C',$VAL(C_i)$ are in normal form and

$C_i = \sqcap_{A \in PRIM(Ci)} A \sqcap \bigsqcap_{C' \in EX(Ci)} \exists r.C' \sqcap \forall r.VAL(C_i)$, $\bot \sqsubset C_i$;

According to the approximation algorithm in [1], the exponential blow-up resulting from interaction between existential and value restrictions occurs immediately at the first level together with the exponential blow-up coming from the $\mathcal{ALC}$-normalization. Therefore, $\mathcal{ALC}$-concept description C should be built such that lcs obtained from computing the approximation at the second level cannot be reduced. With this idea in mind, let now

$A^1_k = \exists r.(P^1_k \sqcap \bigsqcap_{i=1..n, i \neq k}(P^1_i \sqcap P^2_i) \sqcap Q_1 \sqcap Q_2)$,
$A^2_k = \exists r.(P^2_k \sqcap \bigsqcap_{i=1..n, i \neq k}(P^1_i \sqcap P^2_i) \sqcap Q_1 \sqcap Q_2)$ for $k \in \{1, ..., n\}$ and
$B_1 = \exists r.(Q_1 \sqcap \bigsqcap_{i=1..n}(P^1_i \sqcap P^2_i))$,
$B_2 = \exists r.(Q_2 \sqcap \bigsqcap_{i=1..n}(P^1_i \sqcap P^2_i))$ where $P^i_k, Q_j \in N_C$, $r \in N_R$ for $i, j \in \{1, 2\}$, $k \in \{1, ..., n\}$.
Let C be an $\mathcal{ALC}$-concept description as follows :

$C := \exists r.B_1 \sqcap \exists r.B_2 \sqcap \bigsqcap^n_{i=1}(\forall r.A^1_i \sqcup \forall r.A^2_i)$

We must prove that the size of $approx_{ALE}(C)$ is 2^{2^n} and that does not exist any $\mathcal{ALE}$-concept description equivalent to $approx_{ALE}(C)$, the size of which is less than 2^{2^n}.
We have the normal form of C : $C \equiv C_1 \sqcup ... \sqcup C_m$ where $C_i \equiv (\exists r.B_1 \sqcap \exists r.B_2 \sqcap \forall r.val(C_i))$ and $val(C_i) = A^{i_2}_1 \sqcap ... \sqcap A^{i_n}_n$, $(i_1, ..., i_n) \in (\{1, 2\} \times ... \times \{1, 2\})$.
According to the approximation algorithm in [1], $approx_{ALE}(C)$ is computed as follows :

$$approx_{ALE}(C) = \bigsqcap_{(B'_1,...,B'_m) \in (\{B_1,B_2\} \times ... \times \{B_1,B_2\})} \exists r.\text{lcs}\{(B'_i \sqcap val(C_i))|1 \leq i \leq m\} \sqcap \forall r.\text{lcs}\{val(C_i)|1 \leq i \leq m\}$$

Denote $E(B'_1, ..., B'_m)$ as a set of the existential restrictions obtained from the computing of $lcs\{(B'_i \sqcap val(C_i))|1 \leq i \leq m\}$ for each $(B'_1, ..., B'_m)$. Set $E(B'_1, ..., B'_m)$ can be divided into three following groups :

(I) $E_1(B'_1, ..., B'_m)$ is composed of elements that subsume the following existential restrictions : $\exists r.lcs(ex(A^1_k), ex(A^2_k)) \equiv \exists r.(Q_1 \sqcap Q_2 \sqcap \bigsqcap_{l=1, l\neq k}^{n} (P^1_l \sqcap P^2_l))$ for $k \in \{1..n\}$. It is obvious that $\exists r.lcs(ex(A^1_k), ex(A^2_k)) \in E$ for all $k \in \{1..n\}$.

(II) $E_2(B'_1, ..., B'_m)$ is composed of elements that subsume the following existential restrictions : $\exists r.lcs(ex(B_i), ex(B_j)) \equiv \exists r.(\bigsqcap_{k=1}^{n} (P^1_k \sqcap P^2_k))$ if there are B'_i, B'_j which belong to $(B'_1, ..., B'_m)$ and $B'_i \neq B'_j$, or $\exists r.lcs(ex(B_i), ex(B_i)) \equiv \exists r.(Q_i \sqcap \bigsqcap_{k=1}^{n} (P^1_k \sqcap P^2_k))$ for $i \in \{1, 2\}$ if $B'_1 = ... = B'_m$. We also have $\exists r.lcs(ex(B_i), ex(B_j)) \in E$ and $\exists r.lcs(ex(B_i), ex(B_i)) \in E$ for $i \in \{1, 2\}$.

(III) This group is composed of existential restrictions whose expression under existential restriction is *lcs* applying to $ex(B'_k)$ and $ex(A'_1), ex(A'_2), ..., ex(A'_p)$ where $\{A'_1, ..., A'_p\} \subseteq \{A^{i_1}_1, A^{i_2}_2, ..., A^{i_n}_n\}$ for some $k, 1 \leq k \leq m$ and $(i_1, ..., i_n) \in (\{1, 2\} \times ... \times \{1, 2\})$. To determine the relationship between k and $(i_1, ..., i_n)$, we use the following property : for each $(i_1, ..., i_n)$ $\in (\{1, 2\}, ..., \{1, 2\})$ there exists unique $k \in \{1, ..., m\}$ such that $val(C_k)$ does not contain $A^{i_l}_l$ for all $l \in \{1, ..., n\}$. Thus, we denote $\bar{I}(k) = (\bar{i}_1, ..., \bar{i}_n)$ where $val(C_k) = (A^{i_1}_1 \sqcap A^{i_2}_2 \sqcap ... \sqcap A^{i_n}_n)$ and $\bar{i}_h \neq i_h$ for all $h = 1..n$, and $A^{\bar{I}(k)} = \{A^{\bar{i}_1}_1, ..., A^{\bar{i}_n}_n\}$. All elements of this group can be represented as set $E_3(B'_1, ..., B'_m)$ which is composed of the restrictions that subsume the following existential restrictions :
$\exists r.lcs\{ex(B'_k), ex(A'_1), ex(A'_2), ..., ex(A'_p)\}$ for some $\{A'_1, ..., A'_p\} \subseteq A^{\bar{I}(k)}$ where $B'_k \in \{B'_1, ..., B'_m\}$, $B'_k \in ex(C_k)$.
We can prove that for each $(B'_1, ..., B'_m) \in \{(ex(C_1) \times ... \times ex(C_m))\}$, there exists $e' \in E_3(B'_1, ..., B'_m)$ such that $e'' \not\sqsubseteq e'$ for all $e'' \in E(B''_1, ..., B''_m)$ where $(B''_1, ..., B''_m) \in \{(ex(C_1) \times ... \times ex(C_m))\}$ and $(B'_1, ..., B'_m) \neq (B'_1, ..., B'_m)$. According to *Theorem 8* in [3], we can affirm that C has the expected properties. Details of the proof can be found in [9].

6. Conclusion

We have presented a specific data structure for representation of $\mathcal{ALE}$-concept descriptions. This data structure allows us to compute and obtain a compact representation for *lcs*. This compact representation can reduce the size of upper approximation of $\mathcal{ALC}$-concept descriptions in $\mathcal{ALE}$ but does not change the complexity class of the approximation algorithm. To show this, we have constructed an example for which the algorithm returns a concept description whose size is doubly exponential in the size of input.

One of the directions for future work is to extend the method of compact representation to concept descriptions resulting from the computation of approximation with weakened semantics. Furthermore, we will investigate a more general problem involving approximation from a DL L_s to another DL L_d such that the set of constructors of L_d is a subset of the set of constructors of L_s.

References

[1] S. Brandt, R. Küsters and A. Y. Turhan. Approximation and Difference in Description Logics. *Proceeding of KR2002.*

[2] S.Brandt, R. Küsters and A. Y. Turhan. Approximation ALCN-Description Logics. *Proceedings of DL'02.*

[3] F. Baader, R. Küsters and R. Molitor. Computing Least Common Subsumer in Description Logics with Existential Restrictions. *Proceeding of IJCAI*-99.

[4] F. Baader and A. Y. Turhan. On the problem of Computing Small Representations of Least Common Subsumers. *Proceeding of KI2002.*

[5] F. M. Donini, M. Lenzerini, D. Nardi and A. Schaerf. Reasoning in Description Logics. *CSLI Publications* 1997

[6] F. M. Donini, B. Hollunder, M. Lenzerini, A. M. Spaccamela, D. Nardi and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence* 1992.

[7] M. Schmidt-Schaußand G. Smolka. Attribute concept descriptions with complements. *Artificial Intelligence* 1991.

[8] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence, 1990*

[9] Chan Le Duc, Nhan Le Thanh, On the Problems of Computing Approximation and Representing Least Common Subsumer in Description Logics. See http://www.i3s.unice.fr/~cleduc