

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

APPROXIMATION FROM DESCRIPTION LOGICS WITH DISJOINTS TO ANOTHER WITHOUT DISJOINTS

Chan LE DUC, Nhan LE THANH

Projet MECOSI

Rapport de recherche
I3S/RR-2002-23-FR

Juillet 2002

RÉSUMÉ :

Les inférences standards et non-standards dans les Logiques de Description avec la disjonction sont négligemment étudiées tandis qu'il y a beaucoup de recherches sur ces inférences dans les Logiques de Description avec la conjonction. L'inférence d'approximation est citée et étudiée pour la première fois par S. Brandt, R.Küsters et A.Turhan. Les auteurs ont proposé un algorithme double exponentiel pour calculer l'approximation supérieure d'une ALC-description de concept à une ALE-description de concept (ALC-ALE). Un tel algorithme double exponentiel n'est pas satisfaisant dans la pratique. Cet article étudie la subsomption dans le langage ALU(ELU). Un algorithme d'approximation d'une ALU-description de concept à une FL- -description de concept est directement déduit à partir de cette subsomption. Une autre contribution de cet article est de proposer un algorithme exponentiel pour le calcul de l'approximation supérieure ALC-ALE. L'approche choisie pour cet algorithme est le calcul récursif sur la disjonction et la conjonction.

MOTS CLÉS :

Logiques de description, langage ALU, langage ALE, algorithme d'approximation

ABSTRACT:

Standard and non-standard inferences in Description Logics with disjunction are still negligently studied even if these are investigated well in Description Logics with conjunction. Approximation inference is new and first investigated by S. Brandt, R.Küsters and A.Turhan. The authors have proposed a double exponential algorithm computing the upper approximation from an ALC-concept description to an ALE-concept description (ALC-ALE). Such a double exponential algorithm is difficult to be satisfactory in implementation. This paper studies subsumption in ALU (ELU) language. An approximation algorithm from an ALU-concept description to FL- -concept description is deduced directly from this subsumption. The second contribution of the present paper is to propose an exponential algorithm for computing the upper approximation ALC-ALE. The approach chosen for this algorithm is recursive computations on disjunction and conjunction.

KEY WORDS :

Description Logics, ALU language, ALE language, approximation algorithm

Approximation from Description Logics with disjoints to another without disjoints

Chan Le Duc, Nhan Le Thanh

Laboratoire I3S, Université de Nice-Sophia Antipolis

Email: cleduc@i3s.unice.fr, Nhan.Le-Thanh@unice.fr

Résumé

Les inférences standards et non-standards dans les Logiques de Description avec la disjonction sont négligemment étudiées tandis qu'il y a beaucoup de recherches sur ces inférences dans les Logiques de Description avec la conjonction. L'inférence d'approximation est citée et étudiée pour la première fois par S. Brandt, R.Küsters et A.Turhan. Les auteurs ont proposé un algorithme *double exponentiel* pour calculer l'approximation supérieure d'une \mathcal{ALC} -description de concept à une \mathcal{ALE} -description de concept (\mathcal{ALC} - \mathcal{ALE}). Un tel algorithme double exponentiel n'est pas satisfaisant dans la pratique. Cet article étudie la subsomption dans le langage \mathcal{ALU} (\mathcal{ELU}). Un algorithme d'approximation d'une \mathcal{ALU} -description de concept à une \mathcal{FL}^- -description de concept est directement déduit à partir de cette subsomption. Une autre contribution de cet article est de proposer un algorithme *exponentiel* pour le calcul de l'approximation supérieure \mathcal{ALC} - \mathcal{ALE} . L'approche choisie pour cet algorithme est le calcul récursif sur la disjonction et la conjonction.

Abstract

Standard and non-standard inferences in Description Logics with disjunction are still negligently studied even if these are investigated well in Description Logics with conjunction. Approximation inference is new and first investigated by S. Brandt, R.Küsters and A.Turhan. The authors have proposed a *double exponential* algorithm computing the upper approximation from an \mathcal{ALC} -concept description to an \mathcal{ALE} -concept description (\mathcal{ALC} - \mathcal{ALE}). Such a double exponential algorithm is difficult to be satisfactory in implementation. This paper studies subsomption in \mathcal{ALU} (\mathcal{ELU}) language. An approximation algorithm from an \mathcal{ALU} -concept description to \mathcal{FL}^- -concept description is deduced directly from this subsomption. The second contribution of the present paper is to propose an *exponential* algorithm for computing the upper approximation \mathcal{ALC} - \mathcal{ALE} . The approach chosen for this algorithm is recursive computations on disjunction and conjunction.

1 Introduction

This work is motivated by problems automatically translating a knowledge-base written in an expressive DL into another knowledge-base (which uses the same base concepts) in less expressive DL. In electronic commerce, in fact, document interchanges between partners are required. Composition of these documents needs a vocabulary for each partner and a mechanism interpreting a vocabulary into another. One of solutions is to use DLs for designing local vocabularies that are derived from a common, fundamental vocabulary. The vocabularies are designed probably with different DLs. The mechanism interpreting between derived vocabularies, therefore, requires an efficient algorithm allowing to translate a L_s - concept description into L_d -concept description. In the case of a precise translation does not exist, the approximation allows to replace a L_s -concept description with the “nearest” L_d -concept description w.r.t subsumption. Problem becomes interesting if L_d is less expressible than L_s .

We begin it with investigation of subsumption problem in \mathcal{ALU} in section 3. We end this section with an algorithm for computing approximation $\mathcal{ALU}\text{-}\mathcal{FL}^-$. This approximation is based on characterizing subsumption in \mathcal{ALU} .

In section 4, necessary transformations on L_s - concept description are defined for approximation $\mathcal{ALC}\text{-}\mathcal{ALE}$. These transformations allow to compute recursively on disjunction and conjunction. Use of recurrence is one of crucial points that helps avoiding the double exponential complexity in the approximation algorithm.

2 Description Logics

Concept descriptions are built from concept constructors, a set N_C of concept names and a set N_R of role names. The semantics of a concept description is defined in provided that an interpretation $I=(\Delta, \cdot^I)$ as the following table :

Syntax	Semantics	\mathcal{FL}^-	\mathcal{ALU}	\mathcal{ELU}	\mathcal{ALE}	\mathcal{ALC}
\top	Δ	x	x	x	x	x
\perp	\emptyset	x	x	x	x	x
$C \sqcap D$	$C^I \sqcap D^I$	x	x	x	x	x
$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$	x	x		x	x
$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$			x	x	x
$\neg A, A \in N_C$	$\Delta \setminus A^I$	x	x	x	x	x
$C \sqcup D$	$C^I \sqcup D^I$		x	x		x
$\neg C$	$\Delta \setminus C^I$					x

Table 1

In this paper, we study the DLs languages in Table 1.

3 Characterization of subsumption in $\mathcal{ALU}(\mathcal{ELU})$ and $\mathcal{ALU}\text{-}\mathcal{FL}^-$ approximation

We will begin investigate problem of subsumption in $\mathcal{ALU}(\mathcal{ELU})$. The $\mathcal{ALU}\text{-}\mathcal{FL}^-$ approximation is obtained from characterization of this subsumption.

Definition 3.1 Let L_s and L_d be two DLs, and let C be a L_s -concept description and D be a L_d -concept description. Then, D is called upper (lower) L_d -approximation of C ($\min(C)$, $\max(C)$ for short) if
i) $C \sqsubseteq D$ ($D \sqsubseteq C$) and
ii) $C \sqsubseteq D'$ ($D' \sqsubseteq C$), $D' \sqsubseteq D$ ($D \sqsubseteq D'$) implies $D' \equiv D$ for all L_d -concept descriptions D'

Remarks : Let $L_s = \mathcal{ALC}$, $L_d = \mathcal{ALE}$. If there exists $\min(C)$ then it is unique. Indeed, if $D_1 = \min(C)$ and $D_2 = \min(C)$ then $C \sqsubseteq D_1 \sqcap D_2 \sqsubseteq D_1$ and $C \sqsubseteq D_1 \sqcap D_2 \sqsubseteq D_2$. It means that $D_1 \equiv D_2$. There may be many $\max(C)$. For example, $C = \exists r.A \sqcup \forall r.B$, $\max(C) = \exists r.A$, $\max(C) = \forall r.B$.

Definition 3.2 A term, called \forall -normal (\exists -normal) if it is conjunctions of terms of the form :

$$\forall w_i.(A \sqcup B)$$

where A, B are also \forall -normal (\exists -normal) terms. In other word, a normal term is conjunctions of terms of the following form :

$$\forall u_{11}.(\forall u_{21}.(\dots (\forall u_{n1}.(Q_r \sqcup \dots \sqcup Q_s) \sqcup \dots \sqcup \forall u_{nm}.(Q_u \sqcup \dots \sqcup Q_v)) \dots)) \text{ where } Q_i \in \{P_i | P_i \in \mathbb{N}_P\} \cup \{\neg P_i | P_i \in \mathbb{N}_P\}, u_k = r_x \dots r_y \text{ (} r_i : \text{role name)}. (r_i \in \mathbb{N}_R)$$

Remarks : The language \mathcal{ALU} can be \forall -normalized.

Indeed, any term $\forall w.(A \sqcap B)$ can be transformed into the terms : $\forall w.A \sqcap \forall w.B$. The language \mathcal{ELU} can be \exists -normalized.

Definition 3.3 A is a normal term. The suffix word of A , $W(A)$, is defined recursively as follows :

$$\begin{aligned} W(Q) &= Q.\varepsilon, W(Q_1 \sqcup Q_2) = (Q_1, Q_2).\varepsilon, W(\forall u.(Q_1 \sqcup Q_2)) = (Q_1, Q_2).u \\ W(\forall u_1.(A_1 \sqcup A_2)) &= W(A_1).W(A_2).u_1 \\ W(\forall u_1.A_1 \sqcup \forall u_2.A_2) &= W(A_1).u_1.W(A_2).u_2 \end{aligned}$$

Definition 3.4 (Construction of normal term tree)

A is a normal term. The tree of the term A , T_A is defined from $W(A)$ as follow :

1. A node n_0 as root is created.
2. If $W(A)$ is of the form $W(A_1).W(A_2).u$ and assume that two subtrees T_{A_1} and T_{A_2} corresponding to $W(A_1), W(A_2)$ are built. Firstly, two roots of these trees are unified for a unique node n_1 . And then an edge is created in order to connect n_0 with common root n_1 and $(n_0, n_1) = u$. If $W(A)$ is of the form $W(A_1).v.W(A_2).u$ and assume that two subtrees T_{A_1} and T_{A_2} corresponding to $W(A_1), W(A_2)$ are built. Two edges are created in order to connect n_0 with two roots n_{11} and n_{12} : $(n_0, n_{11}) = u, (n_0, n_{12}) = u$.
3. Step 2) is done recursively forward to leaves. And each leaf of tree will be a set $\{Q_i\}$.

Remarks : We denote $P_A(c) = \{Q_i | Q_i \text{ is belong to the leaf } \{Q_i\} \text{ of } c\}$

Examples 3.1

$$A = \forall u_1. (\forall u_2. (P_1 \sqcup P_2) \sqcup (\forall u_3. (P_3 \sqcup P_4)))$$

$$B = \forall u_1. u_2. (P_1 \sqcup P_2) \sqcup \forall u_1. u_3. (P_3 \sqcup P_4)$$

$$C = \forall u_1. (P_1 \sqcup P_2 \sqcup (\forall u_3. (P_3 \sqcup P_4)))$$

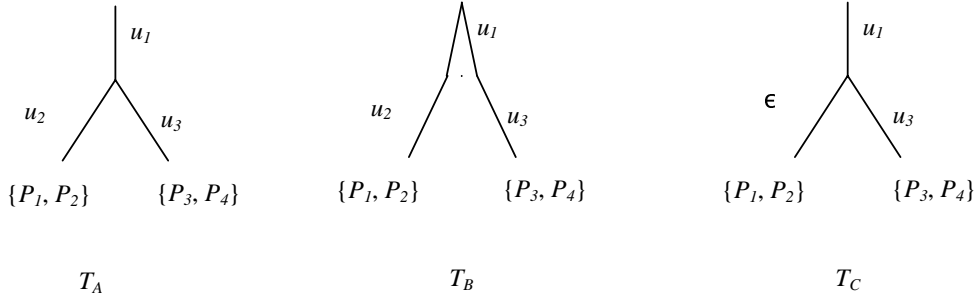


FIG. 1 – Trees T_X

Proposition 3.1 Let T be an (empty) acyclic \mathcal{ALU} -TBox. Let I be a model of T . Let A be concept name occurring in T and be of the form (\forall -normal term).

For any $d \in C^I$ ($C^I = \text{dom}(I)$), we can build a tree $T_d^A \equiv T_A$ i.e d is root of the tree and there exist individuals e_i corresponding to the nodes and leaves n_i of T_A and each edge $(e_i, e_j) = u_{ij}$ of T_d^A corresponds uniquely to an edge $(n_i, n_j) = u_{ij}$ of T_A .

$d \in A^I$ iff

for all trees T_d^A , there exists a path w of the tree T_d^A such that, for all $e \in C^I$: $(d, e) \in w^I \Rightarrow e \in Q_{i_1}^I \vee \dots \vee e \in Q_{i_k}^I$ where $Q_{ij} \in P_A(w)$

A proof of this proposition can be found in Appendix.

Note that a tree for \exists -normal term can be defined similarly (instead there exists a path w , there exists a tree!). Therefore the obtained result for \mathcal{ALU} can be extended for \mathcal{ELU} .

Definition 3.5 Let A_1 and B_1 be normal terms. $W(A_1)$ and $W(B_1)$ are corresponding suffix words.

$W(A_1) \preceq W(B_1)$ is defined as follows :

- If $W(A_1) = \{Q_i\}.w_1$ and $W(B_1) = \{Q_j\}.w_2$ then $w_1 = w_2$ and $P_{A_1}(w_1) \subseteq P_{B_1}(w_2)$
- If $W(A_1) = W(A_2).w_1.W(A_2).w_2$ and $W(B_1) = W(B_1).v_1.W(B_2).v_2$ then for all i there exists j such that $w_i = v_j$ and $W(A_i) \preceq W(B_j)$
- $T_{A_1} \preceq T_{B_1}$ iff $W(A_1) \preceq W(B_1)$

Proposition 3.2 Let T be an empty acyclic \mathcal{ALU} -TBox . Let I be a model of T . Let A_1 and A_2 be \forall -normal terms occurring in T , we have $A_1 \sqsubseteq A_2$ iff $W(A_1) \preceq W(A_2)$

A proof of *Proposition 3.2* can be found in Appendix.

In example 3.1, from Proposition 3.2 we have $B \sqsubseteq A$. *Definition 3.6* allows to extend the obtained result for the language \mathcal{ALU} .

Definition 3.6 Let A_1 and A_2 be normal terms.

- i) $W(A_1 \sqcap A_2) = \{ W(A_1), W(A_2) \}$
- ii) $T_{A_1 \sqcap A_2} = \{ T_{A_1}, T_{A_2} \}$

Theorem 3.1 Let T be an (empty) acyclic T-Box of the language \mathcal{ALU} . Let I be a model of T . Let A_1 and A_2 be concept names occurring in T :

$A_1 = A_{11} \sqcap \dots \sqcap A_{1u}$, $A_2 = A_{21} \sqcap \dots \sqcap A_{2v}$ where A_{ij} are normal terms. We denote :

$$W(A_1) = \{ W(A_{11}), \dots, W(A_{1u}) \}, W(A_2) = \{ W(A_{21}), \dots, W(A_{2v}) \}$$

We have : $A_1 \sqsubseteq A_2$ iff For all $W(A_{2i}) \in W(A_2)$, there exists $W(A_{1j}) \in W(A_1)$ such that

$$W(A_{1j}) \preceq W(A_{2i})$$

A proof of *Theorem 3.1* can be found in Appendix.

Our problem in this section is, however, to find a \mathcal{FL}^- -approximation D of an \mathcal{ALU} -concept description C . Characterizing subsumption in \mathcal{ALU} allows to compute this approximation. *Corollary 3.1* will respond to this problem.

Corollary 3.1 Let C be L_s -concept description where $L_s = \mathcal{ALU}$ and \mathcal{FL}^- -Tbox (it can be empty) as L_d .

There exists a minimal concept description D in L_d

Proof: We denote $T_C = \{T_C^1, \dots, T_C^u\}$ as set of trees where C is considered as disjunctions of C_i terms and T_C^i corresponds to C_i . The same way each word of $L(D, Q_i)$ which is defined in [1] can be considered as tree ended by Q_i . Hence, we can build T_D from words of $L(D, Q_i)$ for all Q_i . Consequently, we obtain a necessary and sufficient condition for the expressivity of C in L_d from *Theorem 3.1*: it says that each T_C^i must be reduced to a path.

Existence of minimal concept description D in L_d such that $C \sqsubseteq D$.

Assume that there exists T_C^i such that T_C^i is a path. We denote $T_C^i \dots T_C^j$ is a set of path trees. In this case, minimal concept description D is built as conjunctions of $D_r = \forall u_1 \dots u_r Q_r$ where $u_1 \dots u_r Q_r = T_C^r$. It is obvious that such a D is minimal. If there does not exist T_C^i such that T_C^i is a path, then $D = \top$.

■

An algorithm can be built directly from *Corollary 3.1* that allows to compute a minimal concept description D in \mathcal{FL}^- from a concept description C in \mathcal{ALU} up to equivalence. This is an algorithm which needs an exponential time to run in the worst case. That is the case if it unfolds all conjunctions and disjoints (normalization).

Example 3.2

- i) $\min_{\mathcal{FL}^-}(A) = \top$ where A is defined in Example 3.1
- ii) $\min_{\mathcal{FL}^-}(A \sqcap \forall r.P) = \forall r.P$

4 \mathcal{ALU} - \mathcal{ALE} (\mathcal{ELU} - \mathcal{ALE}) and \mathcal{ALC} - \mathcal{ALE} approximations

In computation of \mathcal{ALE} -approximation of a concept description containing disjunction we choose another approach. This approach is based on recursive computing of disjunction and conjunction. Indeed, *lcs* computing and normalization allow respectively recurrence on disjunction and conjunction.

Definition 4.1 (the following notations are used in [3]) C is an \mathcal{ALC} -concept description

- $\text{PRIM}^c(C)$ or $\text{PRIM}(C)$ ($\text{PRIM}^d(C)$) denotes the set of all (negated) concept names and the bottom concept occurring on the top-level conjunction (disjunction) of C .
- $\text{VAL}_r^c(C)$ ($\text{VAL}_r^d(C)$ or $\text{VAL}_r(C)$) is a conjunction (set) of all C' occurring in value restrictions of the form $\forall r.C'$ on top-level of C . If there is no value restriction on top-level of C then $\text{VAL}_r^c(C) = \top$.
- $\text{EX}_r^d(C)$ or $\text{EX}_r(C)$ ($\text{EX}_r^c(C)$) is a set (disjunction) of all C' occurring in existential restrictions of the form $\exists r.C'$ on top-level of C .

- The *d-normal* form of C ($d-normal(C)$)
 $C = C_1 \sqcup \dots \sqcup C_m$ where
 $C_i = \sqcap_{A \in PRIM(C_i)} A \sqcap \sqcap_{C' \in EX_r(C_i)} \exists r.C' \sqcap \forall r.VAL_r^c(C_i)$,
 $\perp \sqsubseteq C_i$, C' and $VAL_r^c(C_i)$ are in *d-normal* forms.
- The *c-normal* form of C ($c-normal(C)$)
 $C = C_1 \sqcap \dots \sqcap C_n$ where
 $C_i = \sqcup_{A \in PRIM(D_i)} A \sqcup \exists r.EX_r^d(C_i) \sqcup \sqcup_{C' \in VAL_r(D_i)} \forall r.C'$
 $C_i \sqsubseteq \top$, C' and $EX_r^d(C_i)$ are in *c-normal* forms.

Remarks : C can be turned into an equivalent concept description of the *d-normal* form. C can be turned into an equivalent concept description of the *c-normal* form.

Definition 4.2 Let C be an \mathcal{ALC} -concept description. C is in *C-normal* form if none of the following rules can be applied at any position in C_i :

$$\begin{aligned}
P \sqcap \neg P &\rightarrow \perp \\
E \sqcap \perp &\rightarrow \perp \\
\exists r.\perp &\rightarrow \perp \\
\forall r.\top &\rightarrow \top \\
E \sqcap \top &\rightarrow \top \\
\forall r.C \sqcap \forall r.D &\rightarrow \forall r.(C \sqcap D) \\
\forall r.C \sqcap \exists r.D &\rightarrow \forall r.C \sqcap \exists r.(C \sqcap D)
\end{aligned}$$

where C_i are terms occurring in *d-normal* form of $C = C_1 \sqcup \dots \sqcup C_m$

Definition 4.3 Let C_1, \dots, C_n be the L -concept descriptions. The L -concept description C is the least common subsumer (*lcs*) of C_1, \dots, C_n ($C = lcs(C_1, \dots, C_n)$ for short) iff i) $C_i \sqsubseteq C$ for all $i=1..n$ ii) $C_i \sqsubseteq C'$ for all $i=1..n$ that implies $C \sqsubseteq C'$.

Proposition 4.1

1. If $A = A_1 \sqcap C$ and $B = B_1 \sqcap C$ are normalized \mathcal{ALC} -concept descriptions, we have
 $lcs(A, B) = C \sqcap lcs(A_1, B_1)$
2. If $C = C_1 \sqcap C_2$ is \mathcal{ALC} -concept description which is normalized, we have
 $min_{\mathcal{ALC}}(C) = min_{\mathcal{ALC}}(C_1) \sqcap min_{\mathcal{ALC}}(C_2)$

Proof :

1. Since A and B are *normalized* \mathcal{ALC} -concept descriptions, the description tree of A is an exact concatenation of description tree A_1 and C . Similarly, the description tree of B is an exact concatenation of description tree B_1 and C . From the construction of product description tree for $lcs(A, B)$ in

[5], we have : $lcs(A, B) = lcs(C, C) \sqcap lcs(C, B_1) \sqcap lcs(C, A_1) \sqcap lcs(A_1, B_1)$
 $=$

$C \sqcap lcs(C, B_1) \sqcap lcs(C, A_1) \sqcap lcs(A_1, B_1)$.

It is obvious that : $C \sqcap lcs(C, B_1) \sqcap lcs(C, A_1) \sqcap lcs(A_1, B_1) \sqsubseteq C \sqcap lcs(A_1, B_1)$

Moreover, $C \sqcap lcs(A_1, B_1) \sqsubseteq lcs(C, B_1) \sqcap lcs(C, A_1)$. Hence, $lcs(A, B) = C \sqcap lcs(A_1, B_1)$

2. For short, we write min for $min_{\mathcal{AL}\mathcal{E}}$. We will prove it by induction on the structure.

$C = \forall r. A \sqcap \exists r. (A \sqcap B)$.

We have $min(C) = min(\forall r. A) \sqcap min(\exists r. (A \sqcap B)) = \forall r. A \sqcap \exists r. (A \sqcap B)$. Let $C_i = C_{i1} \sqcup \dots \sqcup C_{ik}$ is d -normal form of C_i . Hence,

$min(C) = min(C_1 \sqcap C_2) = min((C_{11} \sqcup \dots \sqcup C_{1m}) \sqcap (C_{21} \sqcup \dots \sqcup C_{2n})) = min((C_{11} \sqcap C_{21}) \sqcup \dots \sqcup (C_{1m} \sqcap C_{2n})) = lcs(min(C_{11} \sqcap C_{21}), \dots, min(C_{1m} \sqcap C_{2n}))$. By induction hypothesis,

$lcs(min(C_{11} \sqcap C_{21}), \dots, min(C_{1m} \sqcap C_{2n})) = lcs(min(C_{11}) \sqcap min(C_{21}), \dots, min(C_{1m}) \sqcap min(C_{2n}))$

Since C is normalized, the terms $min(C_{1i}) \sqcap min(C_{2j})$ and $min(C_{1i}) \sqcap lcs(min(C_{21}), \dots, min(C_{2n}))$ are normalized too. From 1) we have

$lcs(min(C_{11}) \sqcap min(C_{21}), \dots, min(C_{1m}) \sqcap min(C_{2n})) = lcs(min(C_{11}) \sqcap lcs(min(C_{21}), \dots, min(C_{2n})), \dots, min(C_{1m}) \sqcap lcs(min(C_{21}), \dots, min(C_{2n}))) = lcs(min(C_{21}), \dots, min(C_{2n})) \sqcap lcs(min(C_{11}), \dots, min(C_{1m})) = min(C_1) \sqcap min(C_2)$

■

Theorem 4.1 Let C be an $\mathcal{AL}\mathcal{U}$ ($\mathcal{EL}\mathcal{U}$)-concept description.

1. if $C \equiv \perp$ or $C \equiv \top$ then $min_{\mathcal{AL}\mathcal{E}}(C) = \perp$ or $min_{\mathcal{AL}\mathcal{E}}(C) = \top$
2. if $C = C_1 \sqcup C_2$ where $\perp \sqsubseteq C_1, C_2$ then,
 $min_{\mathcal{AL}\mathcal{E}}(C) = lcs(min_{\mathcal{AL}\mathcal{E}}(C_1), min_{\mathcal{AL}\mathcal{E}}(C_2))$
3. if $C = C_1 \sqcap C_2$ where $C_1, C_2 \sqsubseteq \top$,
 $min_{\mathcal{AL}\mathcal{E}}(C) = min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcap min_{\mathcal{AL}\mathcal{E}}(C_2)$

Proof :

1. Evidently !
2. If C can be written in $C = C_1 \sqcup C_2$ where $\perp \sqsubseteq C_1, C_2$. Suppose that there exists an $\mathcal{AL}\mathcal{E}$ -concept description D such that $C = C_1 \sqcup C_2 \sqsubseteq D \sqsubseteq lcs(min_{\mathcal{AL}\mathcal{E}}(C_1), min_{\mathcal{AL}\mathcal{E}}(C_2))$. If $min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcup min_{\mathcal{AL}\mathcal{E}}(C_2) \sqsubseteq D$ then $D \equiv lcs(min_{\mathcal{AL}\mathcal{E}}(C_1), min_{\mathcal{AL}\mathcal{E}}(C_2))$ (since $min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcup min_{\mathcal{AL}\mathcal{E}}(C_2) \sqsubseteq lcs(min_{\mathcal{AL}\mathcal{E}}(C_1), min_{\mathcal{AL}\mathcal{E}}(C_2))$). If $min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcup min_{\mathcal{AL}\mathcal{E}}(C_2) \not\subseteq D$, there

exists $d \in (\min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcup \min_{\mathcal{AL}\mathcal{E}}(C_2))^I$ and $d \notin D^I$. Hence, we have two possibilities :

- if $d \in (\min_{\mathcal{AL}\mathcal{E}}(C_1))^I$ and $d \notin D^I$: $C_1 \sqsubseteq D \sqcap \min_{\mathcal{AL}\mathcal{E}}(C_1) \sqsubset \min_{\mathcal{AL}\mathcal{E}}(C_1)$ which is a contradiction.
- if $d \in (\min_{\mathcal{AL}\mathcal{E}}(C_2))^I$ and $d \notin D^I$: $C_2 \sqsubseteq D \sqcap \min_{\mathcal{AL}\mathcal{E}}(C_2) \sqsubset \min_{\mathcal{AL}\mathcal{E}}(C_2)$ which is a contradiction.

3. Since $C = C_1 \sqcap C_2$ is an $\mathcal{AL}\mathcal{U}$ ($\mathcal{EL}\mathcal{U}$)-concept description, it is normalized. The proof is obtained immediately from *Proposition 4.1*

Corollary 4.1 The Algorithm $\min_{\mathcal{AL}\mathcal{E}}^{\mathcal{AL}\mathcal{U}}(C)$ is polynomial time algorithm in the size of C where C is an $\mathcal{AL}\mathcal{U}$ ($\mathcal{AL}\mathcal{E}$)-concept description.

Input : $\mathcal{AL}\mathcal{U}$ -concept description C Output : $\min_{\mathcal{AL}\mathcal{E}}(C)$

- If $C = \forall r.C_1$ ($C = \exists r.C_1$) , $\min_{\mathcal{AL}\mathcal{E}}(C) = \forall r.(\min_{\mathcal{AL}\mathcal{E}}(C_1))$ ($\min_{\mathcal{AL}\mathcal{E}}(C) = \exists r.(\min_{\mathcal{AL}\mathcal{E}}(C_1))$)
- If C can be written in $C = C_1 \sqcup C_2$ where $\perp \sqsubset C_1, C_2$,
 $\min_{\mathcal{AL}\mathcal{E}}(C) = lcs(\min_{\mathcal{AL}\mathcal{E}}(C_1), \min_{\mathcal{AL}\mathcal{E}}(C_2))$
- If $C = C_1 \sqcap C_2$ where $C_1, C_2 \sqsubset \top$,
 $\min_{\mathcal{AL}\mathcal{E}}(C) = \min_{\mathcal{AL}\mathcal{E}}(C_1) \sqcap \min_{\mathcal{AL}\mathcal{E}}(C_2)$

Figure 2 : Algorithm for $\mathcal{AL}\mathcal{U}$ ($\mathcal{EL}\mathcal{U}$)- $\mathcal{AL}\mathcal{E}$ approximation

Theorem 4.2 Let C be an $\mathcal{AL}\mathcal{C}$ -concept description.

1. If $C \equiv \perp$ or $C \equiv \top$ then $\min_{\mathcal{AL}\mathcal{E}}(C) = \perp$ or $\min_{\mathcal{AL}\mathcal{E}}(C) = \top$
2. if $C = C_1 \sqcup C_2$ where $\perp \sqsubset C_1, C_2$ then,
 $\min_{\mathcal{AL}\mathcal{E}}(C) = lcs(\min_{\mathcal{AL}\mathcal{E}}(C_1), \min_{\mathcal{AL}\mathcal{E}}(C_2))$
3. if $C = C_1 \sqcap \dots \sqcap C_m$ where $C_i \sqsubset \top$. We have $\min_{\mathcal{AL}\mathcal{E}}(C) = \prod_q \min_{\mathcal{AL}\mathcal{E}}(E_q)$ where the size of E_i is a polynomial of size of C and the number of terms E_i is exponential w.r.t the size of C .

Proof :

1. Evidently !
2. As *Theorem 4.1*.
3. We have the d -formal(C) = $D_1 \sqcup \dots \sqcup D_n$ where
 $D_j = \prod_{A \in PRIM(D_j)} A \sqcap \prod_{C' \in EX(D_j)} \exists r.C' \sqcap \forall r.VAL_r^c(D_j)$. In order to normalize d -formal(C), we replace the terms $\exists r.C'$ with $\exists r.(C' \sqcap VAL_r^c(D_j))$. Next, we compute c -formal(d -normal(C)) = $E_1 \sqcap \dots \sqcap E_m$. We must prove that the size of E_i is a polynomial in size of C and the number of terms E_i is exponential in size of C . Indeed, the size of D_j do not exceed a polynomial

number in the size of C and there are at most an exponential number of D_j . Moreover, let k be the number of atomics : primitive, existential and value restriction terms at top-level of C . The terms E_i are built from at most different $(k + k^2)$ atomic terms (there are at most k^2 new terms $\exists r.(C' \sqcap \text{VAL}_r^c(D_k))$), hence the size E_i does not exceed $(k + k^2)$. The same way, the number of E_i do not exceed q where $q = C_1^{(k+k^2)} + \dots + C_{k+k^2}^{(k+k^2)}$ and each $C_i^{(k+k^2)}$ does not exceed an exponential number in size of C . Hence, q is less than $(k + k^2) \cdot \text{Exp}(C)$. Consequently, from *Proposition 4.1* we have : $\min_{\mathcal{AL}\mathcal{E}}(C) = \min_{\mathcal{AL}\mathcal{E}}(c\text{-normal}(d\text{-normal}(C))) = \prod_1^q \min(E_i)$ where q is an exponential number in size of C and the size of E_i is a polynomial in size of C .

Note that the normalization is done only one time in the computation of $\min_{\mathcal{AL}\mathcal{E}}(C)$ i.e in next recursive steps in which the computing of $\min_{\mathcal{AL}\mathcal{E}}(A_1 \sqcap \dots \sqcap A_m)$ is required, we have immediately $\min_{\mathcal{AL}\mathcal{E}}(A_1 \sqcap \dots \sqcap A_m) = \min_{\mathcal{AL}\mathcal{E}}(A_1) \sqcap \dots \sqcap \min_{\mathcal{AL}\mathcal{E}}(A_m)$. Without normalization is necessary.

■

<p>Input : $\mathcal{AL}\mathcal{C}$-concept description C</p> <p>Output : $\min_{\mathcal{AL}\mathcal{E}}(C)$</p> <ol style="list-style-type: none"> 1. If $C = \forall r.C_1$ ($C = \exists r.C_1$), $\min_{\mathcal{AL}\mathcal{E}}(C) = \forall r.(\min_{\mathcal{AL}\mathcal{E}}(C_1))$ ($\min_{\mathcal{AL}\mathcal{E}}(C) = \exists r.(\min_{\mathcal{AL}\mathcal{E}}(C_1))$) 2. If C can be written in $C = C_1 \sqcup \dots \sqcup C_n$ where $\perp \sqsubset C_i$ $\min_{\mathcal{AL}\mathcal{E}}(C) = \text{lcs}(\min_{\mathcal{AL}\mathcal{E}}(C_1), \dots, \min_{\mathcal{AL}\mathcal{E}}(C_n))$ 3. If C can be written in $C = C_1 \sqcap \dots \sqcap C_n$ where $C_i \sqsubset \top$ <ol style="list-style-type: none"> 3.1 Computing d-normal form of C : $d\text{-normal}(C) = D_1 \sqcup \dots \sqcup D_n$ 3.2 For each D_i, replacing $\exists r.C'$ with $\exists r.(C' \sqcap \text{VAL}_r^c(D_i))$, where $C' \in \text{EX}_r(D_i)$, we obtain $d\text{-normal}(C) = \bigsqcup_i D'_i$ 3.3 Computing c-normal(d-normal(C)) = $\prod_1^q E_i$ 3.4 $\min_{\mathcal{AL}\mathcal{E}}(C) = \prod_1^q \min_{\mathcal{AL}\mathcal{E}}(E_i)$
--

Figure 3 : Algorithm for $\mathcal{AL}\mathcal{C}$ - $\mathcal{AL}\mathcal{E}$ approximation

Corollary 4.2 The algorithm $\min_{\mathcal{AL}\mathcal{E}}^{\mathcal{AL}\mathcal{C}}(C)$ is exponential time algorithm in the size of C where C is an $\mathcal{AL}\mathcal{C}$ -concept description.

Proof :

The *step 1* is done with a polynomial complexity in size of C . In the *step 2*, we have the number of $\min_{\mathcal{AL}\mathcal{E}}(C_i)$ parameters in lcs is polynomial in the size of C . The size of $\min_{\mathcal{AL}\mathcal{E}}(C_i)$ is at most exponential in the size of C . The

$\min_{\mathcal{AL}\mathcal{E}}(C_i)$'s are normalized. According to [5] the computing complexity of lcs in which parameters are normalized is product of parameter sizes. The computing of lcs in the algorithm is done with a polynomial number of normalized parameters and the size of each parameter is exponential at most. Therefore, the computing complexity in *the step 2* is at most exponential time in the size of C .

In the *step 3*, from arguments for step 2, *steps 3.1, 3.2, 3.4* are at most exponential in the size of C . *Steps 3.3* can be also computed in exponential time in the size of C . In fact, we have the number of E_i is at most exponential number in the size of C and the size of E_i is at most polynomial in the size of C (*Theorem 4.2*). In the case in which the recursive calls to *step 2* are necessary, we only need prove that the number of propagated parameters of lcs is always polynomial. Indeed, E_i is disjunction of a polynomial number of conjunctions at top-level, hence lcs is always applied to a polynomial number of parameters. Therefore, the computing complexity in total *step 3* is at most exponential time on the size of C . Finally, the recursive computing in both steps is bound by the depth of tree C . Hence, whole evaluation can be carried out in exponential time on the size of C .

■

Example 4.1 i) $C_2 = \forall r.B \sqcap \exists r.A \sqcap (\exists r.B \sqcup \forall r.B)$

Computing d -normal of $C_2 : (\forall r.B \sqcap \exists r.A \sqcap \exists r.B) \sqcup (\forall r.B \sqcap \exists r.A)$

Normalizing d -normal : $(\forall r.B \sqcap \exists r.(A \sqcap B) \sqcap \exists r.B) \sqcup (\forall r.B \sqcap \exists r.(A \sqcap B))$

Computing c -normal : $\forall r.B \sqcap (\forall r.B \sqcup \exists r.(A \sqcap B)) \sqcap (\exists r.(A \sqcap B) \sqcup \forall r.B) \sqcap \exists r.(A \sqcap B) \sqcap (\exists r.B \sqcup \forall r.B) \sqcap (\exists r.B \sqcup \exists r.(A \sqcap B))$

Computing the approximation :

$\min(C_2) = \min(\forall r.B) \sqcap \min(\forall r.B \sqcup \exists r.(A \sqcap B)) \sqcap \min(\exists r.(A \sqcap B) \sqcup \forall r.B) \sqcap$

$\min(\exists r.(A \sqcap B)) \sqcap \min(\exists r.B \sqcup \forall r.B) \sqcap \min(\exists r.B \sqcup \exists r.(A \sqcap B))$

$= \forall r.B \sqcap lcs(\forall r.B, \exists r.(A \sqcap B)) \sqcap lcs(\exists r.(A \sqcap B), \forall r.B) \sqcap \min(\exists r.(A \sqcap B)) \sqcap$

$lcs(\exists r.B, \forall r.B) \sqcap \min(\exists r.B \sqcup \exists r.(A \sqcap B))$

$= \forall r.B \sqcap \top \sqcap \top \sqcap \exists r.(A \sqcap B) \sqcap \top \sqcap \exists r.B = \forall r.B \sqcap \exists r.(A \sqcap B) \sqcap \exists r.B$

■

5 Conclusion

We have studied a standard inference in \mathcal{ALU} . It is the subsumption inference. This characterizing way can be extended for \mathcal{ELU} . From this inference, we have proposed an exponential algorithm in the worst case approximating an \mathcal{ALU} -concept description with \mathcal{FL}^- -concept description. We have also introduced an efficient algorithm computing upper approximation of an \mathcal{ALC} -concept description with $\mathcal{AL}\mathcal{E}$ -concept description. The complexity of this algorithm is exponential time in the size of the \mathcal{ALC} -concept description. Note that there may

exist *many* lower $\mathcal{AL}\mathcal{E}$ -approximations of an $\mathcal{AL}\mathcal{C}$ -concept description and we can compute it with exponential time. Finally, the use of DLs in the integration of independent vocabularies meets still many challenges. One of these is how to use DLs in representation of contextual information if the translation depends on context. Our work in the future is to research a suitable DL representation for context information.

Références

- [1] Franz Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 1996.
- [2] Franz Baader, Ralf Küster, and Ralf Molitor. Rewriting Concepts Using Terminologies. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning* (KR2000).
- [3] Sebastian Brandt, Ralf Küster, Anni-Yasmin Turhan. Approximation and Difference in Description Logics. *Proceedings of the Eight International Conference on Principles of Knowledge Representation and Reasoning* (KR2002).
- [4] Ralf Küster. Non-standard Inferences in Description Logics. *Thesis, Springer* (2001)
- [5] Franz Baader, Ralf Küster, and Ralf Molitor. Computing least common subsumer in description logics with existential restrictions. *Proceedings of the 17th International Joint Conference Artificial Intelligence* (IJCAI 2001)
- [6] Catriel Beeri, Alon Y. Levy and Marie-Christine Rousset. Rewriting Queries Using Views in Description Logics. *Proceedings of the 16th ACM Symposium on Principles of Database Systems (PODS-97), Tucson* (May 1997)
- [7] F.M Donini, M. Lenzerini, D. Nardi. Reasoning in Description Logics. *CSLI Publications* (1997)
- [8] Bernhard Nebel. Reasoning and revision in hybrid representation systems. *Thesis, Springer-Verlag* (1990)

Appendix

Proposition 2.1 Let T be an (empty) acyclic $\mathcal{AL}\mathcal{U}$ -TBox . Let I be a model of T . Let A be concept name occurring in T and be of the form (*normal term*). For any $d \in C^I$ ($C^I = \text{dom}(I)$), we can build a tree $T_d^A \equiv T_A$ i.e d is root of the tree and there exist individuals e_i corresponding to the nodes and leaves n_i of T_A and each edge $(e_i, e_j) = u_{ij}$ of T_d^A corresponds uniquely to an edge $(n_i, n_j) = u_{ij}$ of T_A .

$d \in A^I$ iff

for all trees T_d^A , there exists a path w of the tree T_d^A such that, for all $e \in C^I$: $(d, e) \in w^I \Rightarrow e \in Q_{i1}^I \vee \dots \vee e \in Q_{ik}^I$ where $Q_{ij} \in P_A(w)$

Proof : We prove the proposition by induction on the numbers of operators \forall and \sqcup of the term A . Let m be the number of operators \forall , let n be the number of operators \sqcup

- $m=1, n=1$
 - Assume that $A = \forall u.(Q_1 \sqcup Q_2)$ and $W(A) = (Q_1, Q_2).u$
 - “ \Rightarrow ” : we suppose that there exists a tree T_d^A that do not satisfy required property. It means that there exists individuals e_1 where $(d, e_1) = u$, $e_1 \notin Q_1^I \wedge e_1 \notin Q_2^I$. This implies that $d \notin A^I$.
 - “ \Leftarrow ” : for all $e_1 \in C^I : (d, e_1) \in u^I$. T_d^A can be built and it has only one edge $(d, e_1) = u$, $W(A) = (Q_1, Q_2).u$. Since, from the property of T_d^A , we have $e_1 \in Q_1^I \vee e_1 \in Q_2^I$. Hence, $d \in A^I$
 - Assume that $A = \forall u.Q_1 \sqcup Q_2$ and $W(A) = Q_1.u.Q_2.\varepsilon$
 - “ \Rightarrow ” : we suppose that there exists a tree T_d^A that do not satisfy required property. This implies that there exists individuals e_1 where $(d, e_1) = u$, $e_1 \notin Q_1^I \wedge d \notin Q_2^I$. It means that that $d \notin A^I$.
 - “ \Leftarrow ” : for all $e_1 \in C^I : (d, e_1) \in u^I$. T_d^A can be built and it has two edges $(d, e_1) = u$ and $(d, d) = \varepsilon$. Since, from the property of T_d^A , we have $e_1 \in Q_1^I \vee d \in Q_2^I$. Hence, $d \in A^I$
- We suppose that the proposition is verified for all normalized terms with $m, n \leq k$.
 - Assume that $A = \forall u.(A_1 \sqcup A_2)$ where the numbers of operators \forall, \sqcup of A_i are less than k and $W(A) = W(A_1).W(A_2).u$
 - “ \Rightarrow ” : we suppose that there exists a tree T_d^A with individuals e_i where $(d, e_1) = u$ and leaf individuals $e_j \notin Q_j^I$. From the construction of T_d^A , the left subtree and the right subtree of T_d^A with common root e_1 are $T_d^{A_1}$ and $T_d^{A_2}$. Hence, from the induction hypothesis, we have $e_1 \notin A_1 \wedge e_1 \notin A_2$. Therefore, $d \notin A^I$.
 - “ \Leftarrow ” : for all $e_1 \in C^I : (d, e_1) \in u^I$, we must prove that $e_1 \in A_1^I \vee e_1 \in A_2^I$. We suppose that T_d^A can be built from e_1 (if T_d^A does not exist, then $d \in A^I$ follows immediately) and there exists a path $\{Q_{ij}\}.w.u$ of T_d^A such that, for all $e \in C^I : (d, e) \in (w.u)^I \Rightarrow e \in Q_{i_1}^I \vee \dots \vee e \in Q_{i_k}^I$ where $Q_{ij} \in P_A(w.u)$. From the construction of T_d^A , the left subtree and the right subtree of T_d^A with common root e_1 are also $T_d^{A_1}$ and $T_d^{A_2}$. In addition, we have $\{Q_{ij}\}.w$ is some path of $T_d^{A_1}$ or $T_d^{A_2}$. From the induction hypothesis, we have : $e_1 \in A_1 \vee e_1 \in A_2$.
 - In case where $A = \forall u.A_1 \sqcup A_2$ where the numbers of operators \forall, \sqcup of A_i are less than k and $W(A) = W(A_1).u.W(A_2).\varepsilon$, our argument is similar.

■

Proposition 3.2 Let T be a acyclic T-Box . Let I be a model of T . Let A_1 and A_2 be *normal terms* occurring in T :

$A_1 \sqsubseteq A_2$ iff $W(A_1) \preceq W(A_2)$

Proof :

“ \Rightarrow ” : Assume that T_{A_2} has m paths : c_1, \dots, c_m . Model I can be defined as follow :

$\text{Dom}(I) = \{d_0, d_{1,1}, \dots, d_{1,m1}, d_{2,1}, \dots, d_{2,m2}, \dots, d_{2m,1}, \dots, d_{2m,2mn}\} \cup \{d_1\}$;
 $P_1 = \{d_0, d_{1,1}, \dots, d_{1,m1}, d_{2,1}, \dots, d_{2,m2}, \dots, d_{m,1}, \dots, d_{m,mn}\}$, $P_2 = \text{Dom}(I)$, $Q = P_2$ for all primitive concept $Q \neq P_1$. The roles : $R^I = \{R_{ij} | r_{k1} \dots r_{k,j1} = c_k\} \cup \{p_s\}$ where $c_k = d_0 d_{k,1} \dots d_{k,mk}$, $c_{2k} = c_k$ for all $k = 1..m$ and $p_s = (d_0 \dots d_1)$; $p_s \neq c_k$ for all $k = 1..m$.

From the construction of T_{A_2} , we have $T_{A_2} = \{c_k.P_1, \text{ for all } k = 1..m\}$ and, from the *Proposition 3.1*, we have $d_0 \notin A_2^I$ (since for each c_k , T_{A_2} has two paths : c_k from d_0 to $d_{k,mk} \in P_1^I$ and c_{2k} from d_0 to $d_{2k,m2k} \in \neg P_1^I$). Assume that $T_{A_1} \not\leq T_{A_2}$, i.e T_{A_1} contains whether c_{2k} , $k=1..m$ or p_s . Since all paths are ended by P_2 , from the *Proposition 2.1*, we have $d_0 \in A_1^I$. It means that we have built a model I such that if $T_{A_1} \not\leq T_{A_2}$ then $A_1^I \not\subseteq A_2^I$.

“ \Leftarrow ” : Assume that $A_1 \not\subseteq A_2$ i.e there exists a model I and an individual $d \in \text{Dom}(I)$ such that $d \in A_1^I \setminus A_2^I$. Assume that $T_{A_1} \not\leq T_{A_2}$. Since $d \notin A_2^I$, *Proposition 2.1* says that there exists $T_{A_2}^d$ and there exist individuals $d_0 \dots d_m$ as leaves of $T_{A_2}^d$ such that $d_i \notin Q_i^I$. But then $T_{A_1} \preceq T_{A_2}$ yields that there exists $T_{A_1}^d$ and there exist individuals $d_i \dots d_j$ ($i, j \leq m$) as leaves of T_{A_1} such that $d_i \notin Q_i^I$ and thus $d \notin A_1^I$ which is a contradiction.

■

Theorem 3.1 Let T be a acyclic T-Box of the language $\mathcal{ALU} = \{\sqcap, \sqcup, \forall\}$. Let I be a model of T . Let A_1 and A_2 be *concept names* occurring in T :

$A_1 = A_{11} \sqcap \dots \sqcap A_{1u}$, $A_2 = A_{21} \sqcap \dots \sqcap A_{2v}$ where A_{ij} are normal terms. We denote :

$W(A_1) = \{W(A_{11}), \dots, W(A_{1u})\}$, $W(A_2) = \{W(A_{21}), \dots, W(A_{2v})\}$

We have : $A_1 \sqsubseteq A_2$ iff for all $W(A_{2i}) \in W(A_2)$, there exists $W(A_{1j}) \in W(A_1)$ such that $W(A_{1j}) \preceq W(A_{2i})$

Proof :

“ \Rightarrow ” : Assume that there exists i such that $T_{A_{2,i}}$ which has m paths : c_1, \dots, c_m and $T_{A_{1,j}} \leq T_{A_{2,i}}$ for all j . Model I can be defined as follow :

$\text{Dom}(I) = \{d_0, d_{1,1}, \dots, d_{1,m1}, d_{2,1}, \dots, d_{2,m2}, \dots, d_{2m,1}, \dots, d_{2m,2mn}\} \cup \{d_1, \dots, d_u\}$;
 $P_1 = \{d_0, d_{1,1}, \dots, d_{1,m1}, d_{2,1}, \dots, d_{2,m2}, \dots, d_{m,1}, \dots, d_{m,mn}\}$, $P_2 = \text{Dom}(I)$, $Q = P_2$ for all primitive concept $Q \neq P_1$. The roles : $R^I = \{R_{ij} | r_{k1} \dots r_{k,j1} = c_k\} \cup \{p_1, \dots, p_u\}$ where

$c_k = d_0 d_{k,1} \dots d_{k,mk}$, $c_{2k} = c_k$ for all $k = 1..m$ and $p_s = (d_0 \dots d_s)$, $s = 1..u$, $p_s \neq c_k$ for all $k = 1..m$.

From the construction of $T_{A_{2,i}}$, we have $T_{A_{2,i}} = \{c_k.P_1, \text{ for all } k = 1..m\}$ and, from the *Proposition 3.1 and 3.2*, we have $d_0 \notin A_{2,i}^I$ (since for each c_k , $T_{A_{2,i}}$ has two paths : c_k from d_0 to $d_{k,mk} \in P_1^I$ and c_{2k} from d_0 to $d_{2k,m2k} \in \neg P_1^I$).

Since $T_{A_{1,j}} \leq T_{A_{2,i}}$, $T_{A_{1,j}}$ contains whether c_{2k} , some $k=1..m$ or p_s , $s=1..u$. Since all paths are ended by P_2 , from *Proposition 2.1, 2.2*, we have $d_0 \in A_{1j}^I$ for all j , hence $d_0 \in \bigcap_j A_{1j}^I$, $j=1..v$. This implies that $d_0 \in A_1$. It means that we have

built a model I such that if $T_{A_1, j} \not\subseteq T_{A_2, i}$ for all $j=1..v$ then $A_1^I \not\subseteq A_2^I$.

“ \Leftarrow ” : *Proposition 3.2*

■