Towards a scalable query rewriting algorithm in presence of value constraints

H. Jaudoin[1], F. Flouvat[2], J.-M. Petit[2], and F. Toumani[3]

[1] University of Rennes, ENSSAT Lannion, IRISA, UMR6074 CNRS, France
[2] University of Lyon, INSA-Lyon, LIRIS, UMR5203 CNRS, F-69621, France
[3] University of Clermont-Ferrand, LIMOS, UMR6158 CNRS, France

Abstract. In this paper, we investigate the problem of query rewriting using views in a hybrid language allowing nominals (i.e., individual names) to occur in intentional descriptions. Of particular interest, restricted form of nominals where individual names refer to simple values enable the specification of value constraints, i.e, sets of allowed values for attributes. Such constraints are very useful in practice enabling, for example, fine-grained description of queries and views in integration systems and thus can be exploited to reduce the query processing cost. We use description logics to formalize the problem of query rewriting using views in presence of value constraints and show that the technique of query rewriting can be used to process queries under the certain answer semantics. We propose a sound and complete query rewriting Bucket-like algorithm. Data mining techniques have been used to favor scalability w.r.t. the number of views. Experiments on synthetic datasets have been conducted.

1 Introduction

This work is motivated by an application in the sustainable land and water management domain[4]. We aim at providing a scalable data integration infrastructure for: (i) sharing and analyzing agricultural practices across heterogeneous agricultural data sources, and (ii) verifying their compliance with respect to national and European government regulations. We adopt a Local As View (LAV) approach [14, 19] to build our data integration system and use query rewriting using views as a technique for answering queries in such a system. The process of query rewriting supplies set of query plans formed of views only that must be further evaluated on data in order to produce correct answers.

In this context, `value constraints` over attributes, i.e., sets of allowed values for those attributes, turn out to be a key feature and have a strong practical interest. Indeed, values constraints enable *fine-grained description* of queries and views in integration systems and can be exploited to reduce the query processing cost. In our application context, various data sources provide views which have identical intentional descriptions (i.e., same structures) but the possible values for certain attributes are different (i.e., different value constraints). For example, two different data sources may store information about cultural parcels that have received pesticide in different years and/or are located at different districts. Views describing these data sources can be informally defined as follows:
$S_1.V_1$: *cultural parcels of district number* 23 *or* 63 *that have received pesticide of category* c_1 *or* c_{18} *or* c_{24}.
$S_2.V_2$: *cultural parcels of district number* 03 *or* 26 *or* 43 *that have received pesticide of category* c_2 *or* c_{15} *or* c_{38}.
Therefore view V_1 of data source S_1 supplies cultural parcels only located in district 23 or 63 and that have received pesticide whose category is only c_1 or c_{18} or c_{24}. Consequently, V_1 cannot return cultural parcels from district 69. In this example, without value constraints over the attributes district number and year, views $S_1.V_1$ and $S_2.V_2$ would be identical. Therefore, the use

[4] This is a collaborative project with a French public research institute whose work focuses on sustainable development in non-urban areas.

of value constraints enables a more accurate description of the content of data sources. Moreover, value constraints can also be very useful to express more precise queries. For example, a *typical* query Q in our application can ask for cultural parcels of only district number 23 or 63 that have received pesticide of category c_{20} or c_{38} only. The user is then interested in a particular region, here the district 23 and 63 only and in a particular set of pesticide categories that are known to be compatible with a culture activity. Turning our attention to query processing techniques, the presence of such constraints provides valuable information to identify when a view is not useful for answering a query. For example, here, the view $S_2.V_2$ cannot supply correct answer to the query Q since $S_2.V_2$ gives only *cultural parcels* in *districts* 03 or 26 or 43. Consequently, capturing and exploiting value constraints may improve the query processing costs in two ways. Firstly, it reduces the number of candidate views to be considered in the query rewriting process. Secondly, it prunes the set of sources accessed to answer a query, thereby reducing the network communication cost.

More generally, value constraints play an important role in various application domains. For example, in Databases Management Systems, they allow to represent enumerated data types - e.g., a marital status is either *married*, *single*, *divorced*, or *widowed*- and then to specify the *value integrity constraints* involved by the views. Moreover, such a kind of constraints allows for incomplete information description [8], that can be very practical in open environments like the web. Indeed sometimes users and administrators of data sources are only able to enumerate possible values of attributes instead of giving its exact value. For example, the user knows that the ages of individuals in his/her databases are either 22 or 23 or 24, and cannot be another value. Therefore, it is impossible to give the right age of individuals but it is possible to give a good idea about their ages. Finally, as mentioned in our motivating example, value constraints are very useful to specify queries of the form: "I seek for individuals whose values on a given attribute cannot be outside of the set of values $\{a_1, ..., a_n\}$". For example, with such a kind of constraints, it is possible to ask for documents dealing with only a list of specific topics, or food prepared with only certain ingredients.

In this paper, we study the problem of rewriting queries using views in presence of value constraints both in the queries and in the views. The problem of rewriting queries using views, intensively investigated during the last decade [24, 14], can be formally stated as follow: given a set of views $\mathcal{V}$ and a query Q, both expressed over a global schema $\mathcal{S}$, the purpose is to reformulate Q into a query expression that uses only the views in $\mathcal{V}$ and is maximally contained in Q. While much has been done on the development of query rewriting algorithms for various classes of languages (conjunctive queries, recursive datalog, description logics) [12, 24, 14, 19], to our knowledge, none has dealt with value constraints. First, it is not possible to reuse algorithms as those proposed in the general framework of conjunctive queries in presence of constants [20] or arithmetic comparisons [2] since conjunctive queries, even with disjunction, allow only for specifying the possible values of an attribute and not for *restricting* the range of an attribute to a given set of values. Second, it is neither possible to exploit existing query rewriting algorithms proposed in the description logics setting (e.g., $\mathcal{ALCNR}$ [7]) since values constraints cannot be correctly simulated in such languages. Indeed, the implicit information on number restrictions intrinsic to the value constraints [5] is likely to be lost and then existing algorithms would lead to incomplete algorithms for the problem of answering queries using views in presence of value constraints.

Hereafter, we investigate the query rewriting problem using a Description Logic (DL) [4] based framework. DLs constitute nowadays one of the most important logic-based knowledge representation formalisms in which problems related to representing and reasoning with value constraints have been deeply studied [15]. In particular, DLs provide two constructors respectively the `OneOf` constructor, noted $\mathcal{O}$, and the universal quantifier constructor $\forall$, that allow for a correct representation of value constraints. The first one enables the enumeration of individuals and then representation of sets of values while the second one restricts the range of a given attribute. For

[5] if an individual checks the constraint $\forall departmentNumber.\{03, 63\}$, then this individual has at most *two* possible values over the attribute $departmentNumber$

example, the description $\forall departmentNumber.\{23, 24, 63\}$ denotes the individuals whose the department number is necessarily restricted to 23 or 24 or 63.

Contributions First, we consider the problem of answering queries in the formal setting of the DLs $\mathcal{ALN}$ augmented with a restricted form of the `OneOf` constructor, noted $\mathcal{O}_v$. We show that query rewriting provides a *sound* and *complete* technique to process queries under the certain answers semantics [1]. Then we propose a query rewriting Bucket-like algorithm based on data mining techniques and hypergraph framework. To our knowledge, this query rewriting algorithm is the first to use data mining implementations to favor scalability of the implementation. Experiments on synthetic datasets have been conducted. They show the feasibility of our proposition.

The remainder of this paper is organized as follows: Section 2 presents the $\mathcal{ALN}(\mathcal{O}_v)$ logic and gives a characterization of subsumption for this logic that is appropriate to deal with our query rewriting problem. Section 3 gives a description of a LAV-mediation system in the $\mathcal{ALN}(\mathcal{O}_v)$ setting and focus on the reduction of the problem of query answering using views to the problem of query rewriting using views in $\mathcal{ALN}(\mathcal{O}_v)$. Section 4 presents how the `Bucket` algorithm has been adapted to get all certain answers to a given query. Section 5 shows how the `iZi` platform [11], that supplies efficient and generic implementations of data mining algorithms, can be used to implement the most costly steps of our Bucket-like algorithm. Finally Section 6 is devoted to implementation and experimentations while Section 7 concludes our paper. Proofs are given in Annex.

2 Preliminaries

Description Logics (DLs) [4] allow to represent domain of interest in terms of *concepts or descriptions* (unary predicates) that characterize subsets of the objects (*individuals*) in the domain, and *roles* (binary predicates) over such a domain. Concepts are denoted by expressions formed by means of special constructors. The various description logics differ from one to another based on the set of constructors they allow. For example, the constructors of the so-called $\mathcal{ALN}$ description logic are: the symbol $\top$ is a concept description which denotes the top concept while the symbol $\bot$ stands for the bottom concept; concept conjunction ($\sqcap$), e.g., the concept description $Parent \sqcap Male$ denotes the set of fathers (i.e., male parents); the value restriction ($\forall R.C$), e.g., the description $\forall child.Male$ denotes the set of individuals whose children are all male; the number restriction constructors ($\geq n\ R$) and ($\leq n\ R$), e.g., the description ($\geq 1\ child$) denotes the set of parents (i.e., individuals having at least one child), while the description ($\leq 1\ leader$) denotes the set of individuals that cannot have more than one leader; the negation restricted to atomic concepts ($\neg A$), e.g., the description $\neg Male$ denotes the class of individuals which are not males.

In this paper, we use the DL $\mathcal{ALN}$ extended with the $\mathcal{O}$ constructor that enables building concepts from a set of enumerated nominals [23, 8]. More precisely, we consider a restricted form of the $\mathcal{O}$ constructor, called $\mathcal{O}_v$ and written $\{o_1, \ldots, o_n\}$, where the o_i's refer to simple values. The obtained language, called $\mathcal{ALN}(\mathcal{O}_v)$ [6], allows descriptions of the form $\forall R.\{o_1, \ldots, o_n\}$, called value constraints, where $\{o_1, \ldots, o_n\}$ denotes a set of values. For example, the concept $\forall maritalStatus.\{MARRIED, SINGLE, DIVORCED, WIDOWED\}$ denotes the set of individuals whose marital status is necessarily *married* or *single* or *divorced* or *widowed*.

$\mathcal{ALN}(\mathcal{O}_v)$ syntax and semantics. Let $\mathcal{C}$ be a set of concept names, $\mathcal{N}$ be a set of value names and let $\mathcal{R}$ be a set of role names. In the spirit of [15], we assume $\mathcal{R}$ divided in two disjointed sets: $\mathcal{R}_c$, which denotes roles whose range is the set of usual individuals, and $\mathcal{R}_v$, which denotes roles whose range is a set of values of $\mathcal{N}$. We also consider that $\top_C$ is the top classical concept while $\top_V$ is the top values concept. Let $A \in \mathcal{C}$, $R \in \mathcal{R}$, $R_C \in \mathcal{R}_C$, $R_V \in \mathcal{R}_V$, n a positive integer and $o_i \in \mathcal{N}$ with $i \in [1, n]$. $\mathcal{ALN}(\mathcal{O}_v)$ concept descriptions are built up by means of the following syntaxic rules:

[6] In this paper, we do not consider other constructors on concrete-valued roles as the fills constructor.

$$C, D \rightarrow A \mid \neg A \mid \top_C \mid \bot \mid$$
$$C \sqcap D \mid$$
$$\forall R_C.C \mid \forall R_V.\{o_1, \ldots, o_n\} \mid \forall R_V.\top_V \mid$$
$$(\leq\ n\ R) \mid (\geq\ n\ R)$$

Semantics of concepts is defined by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set, called *interpretation domain* and $.^{\mathcal{I}}$ is a *interpretation function*. We assume that $\Delta^{\mathcal{I}}$ is divided into two disjointed sets: δ_C the set of individuals in the domain and δ_V, the set of values. Hence we have $\top_C^{\mathcal{I}} = \delta_C$ and $\top_V^{\mathcal{I}} = \delta_V$. A concept is interpreted as a subset of $\Delta^{\mathcal{I}}$. A role is interpreted as a subset of $\delta_C \times \Delta^{\mathcal{I}}$. In other words, values of δ_V cannot have any successor by roles. Values are only authorized in range of $\mathcal{R}_V$ roles. The interpretation $\mathcal{I}$ associates each value $o_i \in \mathcal{N}$ with an element $o_i^{\mathcal{I}} \in \delta_V$ such that $o_i \neq o_j$ implies that ${o_i}^{\mathcal{I}} \neq {o_j}^{\mathcal{I}}$ that is, the mapping respects the Unique Name Assumption (UNA). Furthermore the semantic of an arbitrary concept must verify the following equations:
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
$(\forall R_C.C)^{\mathcal{I}} = \{x \in \delta_C \mid \forall y : (x, y) \in {R_c}^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
$(\forall R_V.\{o_1, \ldots, o_n\})^{\mathcal{I}} = \{x \in \delta_C \mid \forall y : (x, y) \in {R_v}^{\mathcal{I}} \rightarrow y \in \{o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}}\} \subseteq \delta_V\}$
$(\leq nR)^{\mathcal{I}} = \{x \in \delta_C \mid \left|\{y \mid (x, y) \in R^{\mathcal{I}}\}\right| \leq n\}$
$(\geq nR)^{\mathcal{I}} = \{x \in \delta_C \mid \left|\{y \mid (x, y) \in R^{\mathcal{I}}\}\right| \geq n\}$

An interpretation is a *model* for a concept C iff $C^{\mathcal{I}} \neq \emptyset$. A concept is *inconsistent*, i.e., $C \equiv \bot$ iff $C^{\mathcal{I}} = \emptyset$ for all interpretation $\mathcal{I}$.

With respect to this semantics, the notions of subsumption and equivalence are defined as follows. A concept C is *subsumed* by a concept D, noted $C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}\ \forall \mathcal{I}$. A concept C is *equivalent* to a concept D, noted $C \equiv D$, iff $C^{\mathcal{I}} = D^{\mathcal{I}}\ \forall \mathcal{I}$.

Characterizing subsumption in $\mathcal{ALN}(\mathcal{O}_v)$. We present now a normal form description for $\mathcal{ALN}(\mathcal{O}_v)$ concepts and a characterization of subsumption w.r.t. this normal form that are appropriate to deal with the problem of rewriting query using views in presence of value constraints. More precisely, we use a structural approach of subsumption in order to further reduce the research space of query rewriting.

In the sequel, we use the letter P to specify either an atomic concept (A) or its negation ($\neg A$) or a set of values (E) or a number restriction ($(\leq nR)$ or $(\geq nR)$), or $\bot$ concept. The normal form $\widehat{C}$ of a concept C is either the $\top$ concept, or a conjunction (nonempty) of descriptions of the form $\forall R_1.(\ldots \forall R_m.P)$ with $m \geq 0$, where $R_1, \ldots, R_m$ are (not necessarily distinct) roles. The description $\forall R_1 \ldots . \forall R_m.P$ is abbreviated by $\forall R_1 \ldots R_m.P$. $R_1 \ldots R_m$ is considered as a word, noted w, over the alphabet $\mathcal{R}_{\mathcal{C}} \cup \mathcal{R}_{\mathcal{V}}$ of roles. More precisely, $R_1 \ldots R_{m-1}$ is a word over $\mathcal{R}_C^*$ and R_m belongs to $\mathcal{R}_C \cup \mathcal{R}_V$. If $m = 0$, then we have an empty word ϵ, i.e., $\forall \epsilon.P$ is an equivalent notation of P. If w and u are two words of $\mathcal{R}^*$, wu denotes the word obtained by the concatenation of w and u, w being a prefix of wu. Consequently, every concept in $\mathcal{ALN}(\mathcal{O}_v)$ can be expressed in its normal form as a conjunction of concepts of the form $\forall w.P$, called *conjuncts*. In the sequel we use the expression $\forall w.P \in C$ to denote that the normal form of a concept C contains a conjunct of the form $\forall w.P$ in its description.

For the sake of brevity, normalization rules that allow to transform $\mathcal{ALN}(\mathcal{O}_v)$ concepts into their normal forms are omitted. They are described in appendix A page 18.

From the normal form introduced previously, Theorem 1 gives a characterization of subsumption between two concepts in $\mathcal{ALN}(\mathcal{O}_v)$.

Theorem 1 (Subsumption). *Let C, D two concepts, expressed in their normal form. $C \sqsubseteq D$ iff one of the following conditions is verified:*

(1) $C \equiv \bot$ or $D \equiv \top_C$, or
(2) for every $\forall w.P \in D$, we have
(2.a) $\forall w.P \in C$ or,
(2.b) $\forall w.E' \in C$ with $E' \subseteq E$ if $P = E$ or,
(2.c) $\forall w.(\leq kR) \in C$ with $k \leq n$ if $P = \leq nR$ or,

(2.d) $\forall w.R.E \in C$ *with* $|E| \leq n$ *if* $P =\leq nR$ *and* $R \in \mathcal{R}_v$ *or,*
(2.e) $\forall w.(\geq kR) \in C$ *with* $k \geq n$ *if* $P =\geq nR$ *or,*
(2.f) $\forall v.(\leq 0R) \in C$ *with* vR *prefix of* w.

Informally, a concept C is subsumed by a concept D if and only if all constraints over D appears in the description of C [7]. Note that we abbreviate the concepts $\forall R.\bot$ and $\forall R.\emptyset$ by $\leq 0R$. The proof of this theorem which is given in annex A, page 18, is derived from characterization of structural subsumption of CLASSIC [8]. Indeed, logic $\mathcal{ALN}(\mathcal{O}_v)$ can be seen as a sub-language of CLASSIC [8] where constructor $\mathcal{O}_v$ can be considered as a particular case of *Host Individuals*.

Example 1. Let $C \equiv \forall received.Pesticide \sqcap CulturalParcel$, $C' \equiv \forall received.category.\{C_2, C_3\} \sqcap \forall received. \leq 2\, category$, $D \equiv \forall received.Pesticide$ and $D' \equiv \forall received.category.\{C_1, C_2, C_3\} \sqcap \forall received. \leq 3\, category$

We have $C \sqsubseteq D$ since $\forall received.Pesticide \in C$. We have $C' \sqsubseteq D'$ because $\forall received.category.\{C_2, C_3\} \in C'$ with $\{C_2, C_3\} \subseteq \{C_1, C_2, C_3\}$, and $\forall received.(\leq 2\, category) \in C'$.

3 Query rewriting using views in the $\mathcal{ALN}(\mathcal{O}_v)$ setting

In this section, we briefly introduce the $\mathcal{ALN}(\mathcal{O}_v)$-based framework used in our work. Then we focus on the problem of *query answering using views*, i.e., computing the answers of a query in presence of value constraints. In this setting, we show that the problem of query answering in the $\mathcal{ALN}(\mathcal{O}_v)$ setting can be reduced to the problem of query rewriting using views.

3.1 A formal framework

A LAV-based mediation system is defined by a pair $(\mathcal{S}, \mathcal{V})$, where $\mathcal{S}$ is a global schema and $\mathcal{V}$, a set of views, i.e., named queries, expressed in terms of $\mathcal{S}$ [14]. Hereafter, we consider a mediation system $(\mathcal{S}, \mathcal{V})$ in the $\mathcal{ALN}(\mathcal{O}_v)$ setting. Therefore, the schema $\mathcal{S}$ is specified as an $\mathcal{ALN}(\mathcal{O}_v)$-terminology, i.e., a set of axioms of the forms: (i) $A \equiv D$ (concept definitions), or (ii) $A \sqsubseteq D$ (primitive specifications), where A is a concept name and D is a concept description in $\mathcal{ALN}(\mathcal{O}_v)$. Moreover, the set of views $\mathcal{V}$ is specified as a set of primitive specifications in $\mathcal{ALN}(\mathcal{O}_v)$.

The semantic of a mediation system $(\mathcal{S}, \mathcal{V})$ is derived from the notion of interpretation of a terminology in DL [4]. We say that an interpretation $\mathcal{I}$ is a model for $(\mathcal{S}, \mathcal{V})$ iff $\mathcal{I}$ is a model for every axiom in $\mathcal{S}$ and $\mathcal{V}$ (i.e., $A \sqsubseteq D$ iff $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $A \equiv D$ iff $A^{\mathcal{I}} = D^{\mathcal{I}}$). Note that, describing views as primitive specifications enables to capture Open World Assumption (OWA) in building our mediation system [1] (i.e., assuming that the data sources are incomplete). Indeed, primitive specifications are incomplete specifications in the sense that they provide only necessary, but not sufficient, conditions that must be satisfied by their instances.

The Table 1 gives an example of $\mathcal{ALN}(\mathcal{O}_v)$ mediation system $(\mathcal{S}, \mathcal{V})$. The global schema is made of two concepts: $CulturalParcel$, which denotes parcels that have at least one kind of culture and $TreatedObject$ which denotes the class of individuals that have received at least one treatment which has at least one category and whose categories take their values necessarily from the set $\{C_1, \ldots, C_{18}\}$. The terminology $\mathcal{V}$ is made of nine views (V_1, V_2, ..., V_9). The *extension* of the view V_1 is a subset of cultural parcels while extension of view V_2 is a subset of individuals that have received at least one category of treatment.

Queries are defined as $\mathcal{ALN}(\mathcal{O}_v)$ concepts expressed in terms of the elements (i.e., roles and concepts) of $\mathcal{S}$. For example, a query Q that asks for cultural parcels that have received at least one treatment and whose treatment category is either C_8 or C_9, may be expressed as follows: $Q \equiv CulturaParcel \sqcap \geq 1 treatment \sqcap \forall treatment.category.\{C_8, C_9\}$.

Let $(\mathcal{S}, \mathcal{V})$ be a $\mathcal{ALN}(\mathcal{O}_v)$ mediation system. In the sequel, we assume that the terminologies $\mathcal{S}$ and $\mathcal{V}$ are acyclic (i.e., they do not contain a concept that refers to itself in its specification or

[7] Following the object-oriented paradigm, C must override the concept D.

Global schema $\mathcal{S}$	
$CulturalParcel \equiv\ Parcel \sqcap \geq\ 1\, cultureType$	
$TreatedObject \equiv \geq 1 treatment \sqcap \forall treatment.\geq 1 category \sqcap \forall treatment.category.\{C_1, C_2, \ldots, C_{18}\}$	
$OrganicallyTreatedObject \equiv \forall treatment.OrganicProduct$	
Set of views $\mathcal{V}$	
$V_1 \sqsubseteq\ CulturalParcel$	$V_4 \sqsubseteq \forall treatment.category.\{C_9, C_{10}\}$
$V_2 \sqsubseteq\ \forall treatment.\geq\ 1 category$	$V_5 \sqsubseteq \forall treatment.category.\{C_8\}$
$V_3 \sqsubseteq \geq\ 1 treatment$	$V_6 \sqsubseteq \forall treatment.category.\{C_8, C_{11}\}$
$V_7 \sqsubseteq \forall treatment.category.\{C_7, C_8, C_9, C_{10}\} \sqcap \forall treatment.OrganicProduct$	
$V_8 \sqsubseteq \forall treatment.category.\{C_7, C_8, C_9\} \sqcap \forall treatment.\neg OrganicProduct$	
$V_9 \sqsubseteq \forall treatment.category.\{C_8, C_9, C_{10}\} \sqcap \forall treatment.\neg OrganicProduct$	

Table 1. Example of mediation system

definition). We also assume that $\mathcal{V}$ is: (i) *normalized*, i.e., every primitive specification $A \sqsubseteq D$ in $\mathcal{V}$ is replaced by a definition $A \equiv \overline{A} \sqcap D$, where $\overline{A}$ is a new atomic concept [4], and (ii) *expanded*, i.e., defined concepts occurring in right-hand side of axioms are recursively replaced by their definitions. Finally, queries on $(\mathcal{S}, \mathcal{V})$ and views in $\mathcal{V}$ are assumed to be provided in their normal forms.

3.2 From query answering to query rewriting

This section focuses on the query answering problem in the $\mathcal{ALN}(\mathcal{O}_v)$ setting. Let Q be a query over a $\mathcal{ALN}(\mathcal{O}_v)$ mediation system $(\mathcal{S}, \mathcal{V})$. We consider the problem of computing the answers of Q under *certain answer* semantics [1]. Informally, an answer t is a certain answer of Q if t is an answer to Q for any database consistent with the extensions of the views in $\mathcal{V}$, i.e., the set of tuples associated with the views. We use the following notations to define the notion of certain answers under OWA in the DL setting. For a view $V \in \mathcal{V}$, we denote by V^{ext} its extension and by $\mathcal{V}^{ext}$ the union of all the extensions of the views in $\mathcal{V}$.

Definition 1 (Certain answers under OWA). *Let $(\mathcal{S}, \mathcal{V})$ be a $\mathcal{ALN}(\mathcal{O}_v)$ mediation system and Q be a query. t is a certain answer of Q iff:* ***(i)*** *$t \in \mathcal{V}^{ext}$ and* ***(ii)*** *$t \in Q^{\mathcal{I}}$, for all model $\mathcal{I}$ of $(\mathcal{S}, \mathcal{V})$ s.t. $\forall\, V \in \mathcal{V}$, $V^{ext} \subseteq V^{\mathcal{I}}$.*

The set of all the certain answers of Q is denoted by $Ans(Q, \mathcal{V}^{ext})$.

Let Q be a query over a mediation system $(\mathcal{S}, \mathcal{V})$. The problem of computing $Ans(Q, \mathcal{V}^{ext})$ can be reduced to a problem of *query rewriting using views* [14, 12]. In the latter case, the goal is to reformulate Q into an expression Q' in some language, denoted $\mathcal{L}_{\mathcal{R}}$, such that Q' refers only to the views of $\mathcal{V}$ and $Q' \sqsubseteq Q$. Q', the rewriting of Q, can be viewed as a query plan in the sense that it can be evaluated over the view extensions in order to compute the certain answers of Q. A crucial point to guarantee that a rewriting Q' provides *all* the certain answers of Q lies in the definition of the rewriting language $\mathcal{L}_{\mathcal{R}}$. Below, we show that in the setting of our $\mathcal{ALN}(\mathcal{O}_v)$ mediation system it is sufficient to consider rewritings that consist in union of view conjunctions (i.e., $\mathcal{L}_{\mathcal{R}} = \{\sqcap, \sqcup\}$).

Lemma 1. *Let $(\mathcal{S}, \mathcal{V})$ be a $\mathcal{ALN}(\mathcal{O}_v)$ mediation system and Q be a query. If t is a certain answer of Q, then there exists a subset $\{V_1, \ldots, V_n\}$ of $\mathcal{V}$ s.t.:* ***(i)*** *$V_1 \sqcap \ldots \sqcap V_n \sqsubseteq Q$, and* ***(ii)*** *$t \in V_1^{ext} \cap \ldots \cap V_n^{ext}$.*

This lemma states that a certain answer of a query Q in a $\mathcal{ALN}(\mathcal{O}_v)$ mediation system is provided by a conjunction of views which is subsumed by Q. Its proof is given in annex B, page 19. Note that the rewriting language obtained in our context is the same than those usually obtained in other modeling languages as for example $\mathcal{ALCNR}$ and CARIN-$\mathcal{ALN}$ [7] or conjunctive queries [20]. Hereafter, we use the notion of *conjunctive rewriting* of a query Q to refer a conjunction of views subsumed by Q. As a consequence of lemma 1, all the certain answers of a query Q can be obtained from the union of all the conjunctive rewritings of Q. We define below the notion of *maximally-contained rewriting*, i.e., those conjunctive rewritings that return maximal sets of certain answers.

Definition 2 (Max-contained rewriting). *Let $(\mathcal{S}, \mathcal{V})$ be a mediation system and Q be a query. Q' is a maximally-contained rewriting of Q using $\mathcal{V}$ if and only if:* ***(i)*** *Q' is a conjunctive rewriting of Q, and* ***(ii)*** *there is no conjunctive rewriting Q_1 of Q s.t. $Q' \sqsubseteq Q_1 \sqsubseteq Q$ and $Q' \not\equiv Q_1$.*

The following theorem shows that the set of *all* certain answers of a query Q can be computed exclusively from the union of the maximally-contained rewritings of Q.

Theorem 2. *Let $(\mathcal{S}, \mathcal{V})$ be a mediation system and Q be a query. Let $\{Q_1, \ldots, Q_n\}$ be the set of all maximally-contained rewritings of Q using $\mathcal{V}$ and $Q_i(\mathcal{V}^{ext})$ the result of evaluating Q_i over $\mathcal{V}^{ext}$.*

$Ans(Q, \mathcal{V}^{ext}) = \cup_{i=1}^{n} Q_i(\mathcal{V}^{ext})$.

Therefore, to compute $Ans(Q, \mathcal{V}^{ext})$, one can restrict our attention to the problem of computing all the maximally-contained rewritings of Q using $\mathcal{V}$. Proof of this theorem is given in annex B, page 19.

To characterize the maximally-contained rewritings of Q and then compute them, we use the interesting following property that is a direct consequence of the open word assumption.

Lemma 2. *Let $\{V_1, \ldots, V_n\}$ be a subset of $\mathcal{V}$ and $Q' \equiv \sqcap_{i=1}^{n} V_i$ s.t. $Q' \sqsubseteq Q$. Q' is a maximally-contained rewriting of Q using views $\mathcal{V}$ iff for any concept Q'' obtained by removing from Q' one of its conjuncts V_i, we have $Q'' \not\sqsubseteq Q$.*

It turns out that any maximally-contained rewritings of Q is necessarily made of a minimal subset of $\mathcal{V}$ such that the conjunction of its elements is subsumed by Q. Proof of the lemma is given in annex B, page 19. Hereafter, the problem of computing $Ans(Q, \mathcal{V}^{ext})$ is then equivalent to the problem, denoted by *conj_rewrite(Q,$\mathcal{V}$,$\mathcal{ALN}(\mathcal{O}_v)$)*, of enumerating all the minimal subsets of $\mathcal{V}$ s.t. the conjunction of their elements is subsumed by Q. Next section gives an algorithm to solve *conj_rewrite(Q,$\mathcal{V}$,$\mathcal{ALN}(\mathcal{O}_v)$)*, thereby providing a sound and complete procedure for our query answering using views problem.

4 A `Bucket`-based algorithm for $\mathcal{ALN}(\mathcal{O}_v)$ mediation system

In the setting of $\mathcal{ALN}(\mathcal{O}_v)$, the optimizations of the `Minicon` algorithm [22] can't be used to compute maximally-contained rewritings, since views and queries are specified as conjunction of *unary* concepts. A possible solution is to follow a "bucket-like approach" [14]. The interest of this approach is to break down the problem of rewriting maximally a query into rewriting maximally each of its subgoals, here the query conjuncts. The algorithm 1, called `ComputeRew`, is a slight adaptation of the `Bucket` algorithm [14].

Algorithm 1 ComputeRew

Require: $\mathcal{V} = \{V_1, ..., V_m\}$ a set of views and Q a query
Ensure: $\mathcal{MCR}$ the set of maximally-contained rewriting of Q using $\mathcal{V}$
1: Let $Q \equiv \sqcap_{i=1}^{n} \forall w_i.P_i$
2: **/* Step 1: Buckets computation */**
3: **for all** conjunct $\forall w_i.P_i$ **do**
4: $B(w_i, P_i)$ = BucketBuilding($\mathcal{V}$,$\forall w_i.P_i$)
5: */* Pruning of inconsistent and non maximal rewritings */*
6: $B(w_i, P_i)$:=BucketPruning($B(w_i, P_i)$)
7: **end for**
8: **/* Step 2: Rewritings generation */**
9: $\mathcal{MCR}$:=Cart_Prod($B(w_i, P_i)$, $i \in \{1, ..., n\}$)
10: */* Pruning of inconsistent and non maximal rewritings */*
11: $\mathcal{MCR}$:=Pruning($\mathcal{MCR}$)
12: RETURN $\mathcal{MCR}$

Given a rewriting problem *conj_rewrite(Q,$\mathcal{V}$,$\mathcal{ALN}(\mathcal{O}_v)$)* with $Q \equiv \forall w_1.P_1 \sqcap \ldots \sqcap \forall w_n.P_n$, the algorithm `ComputeRew`, as the well-known `Bucket` algorithm is made up of two main steps:

- **Buckets Computation.** For each conjunct $\forall w_i.P_i$ of Q, a *bucket*, noted $B(w_i, P_i)$, containing all the maximally-contained rewritings of this conjunct is created.
- **Rewritings Generation.** This step computes maximally-contained rewritings of Q, denoted by $\mathcal{MCR}$, by combining elements from the previously identified buckets. $\mathcal{MCR}$ is the solution to the problem *conj_rewrite(Q,$\mathcal{V}$,$\mathcal{ALN}(\mathcal{O}_v)$)*.

In the $\mathcal{ALN}(\mathcal{O}_v)$ setting, the step of Buckets computation, i.e., the step 1, must be redefined as detailed in subsection 4.1.

4.1 Bucket algorithm for $\mathcal{ALN}(\mathcal{O}_v)$

Using a case based analysis for the language $\mathcal{ALN}(\mathcal{O}_v)$, the next lemma gives necessary conditions that should be verified by a bucket element (i.e., a rewriting of a conjunct).

Lemma 3. *For conj_rewrite(Q,$\mathcal{V}$,$\mathcal{ALN}(\mathcal{O}_v)$), let $Q \equiv \forall w.P$, l be the cardinality of the largest set of values that appears in $\mathcal{V}$ or Q, and p be the maximal depth*[8] *of the conjuncts in $\mathcal{V}$ or Q. $Q' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_k}$ is a maximally-contained rewriting of Q if Q' is made of a **minimal** subset of $\mathcal{V}$ verifying one of the following conditions:*

a) *$P \in \{A, \neg A\}$ then $\forall w.P \in Q'$ and $k = 1$ or,*

b) *$P = (\geq nR)$ then $\forall w.(\geq pR) \in Q'$ with $p \geq n$ and $k = 1$ or,*

c) *$P = (\leq nR)$ then $\forall w.(\leq pR) \in Q'$ with $p \leq n$ and $k = 1$ or,*

d) *$P = E$ then $\{V_{i_1}, ..., V_{i_k}\}$ is s.t. **(i)** for each $j \in [1, k]$, $\forall w.E_{i_j} \in V_{i_j}$, and $\cap_{j=i_1}^{i_k} E_j \subseteq E$ and **(ii)** $1 \leq k \leq l + 1$ or,*

e) *$P = (\leq\ n\ R_v)$, with $R_v \in \mathcal{R}_v$ then $\{V_{i_1}, ..., V_{i_k}\}$ is s.t. **(i)** for $j \in [1, k]$, $\forall w.E_{i_j} \in V_{i_j}$ and $|\cap_{j=i_1}^{i_k} E_j| \leq n$ **(ii)** $1 \leq k \leq l + 1$ or,*

f) *$\forall w'.(\leq 0v) \in Q'$ with $w'v$ a prefix of w s.t. and $1 \leq k \leq l + p$.*

Proof of this lemma is given in annex C.1, page 21.

The algorithm 2 computes the bucket elements based on this lemma. To the best of our knowledge, algorithm 2 is the first adaptation of the bucket algorithm in the setting of $\mathcal{ALN}(\mathcal{O}_v)$. In this algorithm, we denote by $2^{\mathcal{V}}$ the powerset of $\mathcal{V}$ and by $min_{\subseteq}(S)$ s.t. $S \subseteq 2^{\mathcal{V}}$, the subsets of S that are minimal w.r.t. the set inclusion.

In this context, we can distinguish two types of rewritings: classical $\mathcal{ALN}$ rewritings [12], lines 2-14 of algorithm 2, and specific rewritings due to the presence of value constraints, lines 15-29 of algorithm 2. Note that classical $\mathcal{ALN}$ rewritings are made of only one view and correpond to cases *a*, *b*, *c* of Lemma 3. Rewritings $RewS_1(E, w)$ are due to value constraints while rewritings $RewS_2(n, w.R_v)$ are due to the interaction between the value constraints and number restrictions constructors. Indeed a number restriction can subsume a value constraint as built up by the case (2.d) of Theorem 1. Note that each rewriting in $RewS_1(E, w)$ and in $RewS_2(n, w.R_v)$ consists of conjunction of views such that each view has a value constraint over the word w and respectively over $w.R_v$. Moreover, consequently to lemma 2, such rewritings are made of minimal subsets of views w.r.t. set inclusion, s.t. their conjunction is subsumed by $\forall w.E$, respectively by $\forall w. \leq nR_v$. Therefore to compute $RewS_1(E, w)$ and $RewS_2(n, w.R_v)$, first we must find views having value constraints over w, respectively $w.R_v$. Second, from this set of views, we have to compute the minimal subsets of views $S_1(E, w)$ and $S_2(n, w.R_v)$. A set of views is in $S_1(E, w)$ if the intersection of their value constraints is a subset of E. A set of views is in $S_2(n, w.R_v)$ if the cardinality of the intersection of their value constraints is less than n. At last, the rewritings $RewS_1(E, w)$ and $RewS_2(n, w.R_v)$ are inferred by conjunction of the views belonging to each element of $S_1(E, w)$ and $S_2(n, w.R_v)$. The maximal number of views occurring in such rewritings is $l+1$, the cardinality of the largest set of values occurring in the views $\mathcal{V}$. Finally, the algorithm (computation of II, lines

[8] The depth of a conjunct $\forall w.P$ is equal to the length of the word w.

Algorithm 2 Bucket building

Require: $\mathcal{V} = \{V_1, ..., V_m\}$ a set of views and $\forall w.P$ a conjunct of Q
Ensure: $B(w, P)$

$B(w, P) := \emptyset$
/* Computation of classical $\mathcal{ALN}$ rewritings */
/* Condition a) of lemma 3 */
if $P = A$ or $P = \neg A$ **then**
  $B(w, P) := B(w, P) \cup \{V_i \in \mathcal{V} \mid \forall w.P \in V_i\}$
end if
/* Condition b) of lemma 3 */
if $P = (\geq nR)$ **then**
  $B(w, P) := B(w, P) \cup \{V_i \in \mathcal{V} \mid \forall w.(\geq pR) \in V_i, p \geq n\}$
end if
/* Condition c) of lemma 3 */
if $P = (\leq nR)$ **then**
  $B(w, P) := B(w, P) \cup \{V_i \in \mathcal{V} \mid \forall w.(\leq pR) \in V_i, p \leq n\}$
end if
/* Computation of specific $\mathcal{ALN}(\mathcal{O}_v)$ rewritings */
/* Condition d) of lemma 3 */
if $P = E$ **then**
  /* Computation of the rewritings $RewS_1(E, w)$ */
  $S_1(E, w) = min_{\subseteq}(U \in 2^{\mathcal{V}} \mid \forall V_i \in U, \forall w.E_i \in V_i$ and $\bigcap_{V_i \in U} E_i \subseteq E)$
  $RewS_1(E, w) = \{\sqcap_{V_i \in U} V_i \mid U \in S_1(E, w)\}$
  $B(w, P) := B(w, P) \cup RewS_1(E, w)$
end if
/* Condition e) of lemma 3 */
if $P = (\leq n\, R_v), R_v \in \mathcal{R}_v$ **then**
  /* Computation of the rewritings $RewS_2(n, w.R_v)$*/
  $S_2(n, w.R_v) = min_{\subseteq}(U \in 2^{\mathcal{V}} \mid \forall V_i \in U, \forall w.R_v.E_i \in V_i$ and $|\bigcap_{V_i \in U} E_i| \leq n)$
  $RewS_2(n, w.R_v) = \{\sqcap_{V_i \in U} V_i \mid U \in S_2(n, w.R_v)\}$
  $B(w, P) := B(w, P) \cup RewS_2(n, w.R_v)$
end if
/* Computation of both classical $\mathcal{ALN}$ and specific $\mathcal{ALN}(\mathcal{O}_v)$ implicit inconsistencies*/
/* Condition f) of lemma 3 */
$II = min_{\subseteq}(U \in 2^{\mathcal{V}} \mid \forall w'.(\leq 0v) \in Q' \equiv \sqcap (V_i \in U)$ and w'v is a prefix of w)
$B(w, P) := B(w, P) \cup \{\sqcap_{V_i \in U} V_i \mid U \in II\}$

25-27) processes $\mathcal{ALN}(\mathcal{O}_v)$ *implicit inconsistencies* [18] as built up by the case f of Lemma 3. These implicit inconsistencies are computed from $RewS_2(n, w.R_v)$ and classical implicit inconsistencies. More precisely, for each view $V_i \in \mathcal{V}$ such that $\forall w. \geq mR_v \in V_i$, we must compute $RewS_2(n, w.R_v)$ with $n < m$.

The following example illustrates the bucket building algorithm in our setting.

Example 2. Continuing the example given in Table 1, let us considering the following query made up of three conjuncts:

$Q \equiv CulturalParcel \sqcap \forall treatment.category.\{C_8, C_9\} \sqcap \geq 1 treatment.$

By applying the previous algorithm on the 9 views of the mediator given in Table 1, we get:

- $B(\epsilon, CulturalParcel) = \{V_1\}$ (case **(a)**)
- $B(\epsilon, \geq\ 1\, treatment) = \{V_3\}$ (case **(b)**)
- $B(treatment.category, \{C_8, C_9\}) = \{V_5, V_4 \sqcap V_6, V_7 \sqcap V_8 \sqcap V_9, V_7 \sqcap V_8\}$. The three first rewritings are due to the case **(d)** while the last one is due to case **(f)**.

To end up, note that the obtained buckets need to be pruned in order to remove inconsistent or not maximal rewritings (see line 6 of algorithm 1), which is not the case of the classical `Bucket` algorithm. Indeed implicit inconsistencies may appear in the rewritings, as shown in the following example.

Example 3. Continuing the example 2, the rewriting $V_7 \sqcap V_8 \sqcap V_9$ of the bucket $B(treatment.category, \{C_8, C_9\})$ is not maximal because $V_7 \sqcap V_8$, that infers an implicit inconsitency over "*treatment*" role , belongs to the same bucket. Therefore the rewriting $V_7 \sqcap V_8 \sqcap V_9$ must be deleted from the bucket $B(treatment.category, \{C_8, C_9\})$.

4.2 Max-rewritings generation

The second step of the algorithm 1 constructs the global rewritings of a query (i.e., the set $\mathcal{MCR}$) using the buckets generated previously. In the classical approach, it begins by generating candidate rewritings from the *cartesian product* of the buckets. However, the cost of the cartesian product is prohibitive even on medium size configuration. To cope with this limitation, we propose a new hypergraph-based characterization to avoid the use of costly cartesian product. Indeed, computation of the rewritings can be reduced to a well known problem in combinatorics, the computation of minimal transversals of a hypergraph [10].

Definition 3 (Hypergraph). *Let V be a set of vertices and E an element of the powerset $2^{|V|}$ of V.*

A hypergraph $\mathcal{H} = (V, E)$ consists of a finite collection E of sets over a finite set V. The elements of V are called the vertices of $\mathcal{H}$ while the elements of E are called the edges of $\mathcal{H}$.

Definition 4 (Transversal and minimal transversal). *Let V be a set of vertices and E an element of the powerset $2^{|V|}$ of V.*

$T \subseteq V$ is a transversal of $\mathcal{H}$ if $\forall X \in H, T \cap X \neq \emptyset$.

T is minimal if $\forall Y \subset T, Y$ is not a transversal.

Even if the best complexity of the problem of computing minimal transversal of a hypergraph is known to be in almost-polynomial time [17], efficient and scalable implementations exist since this problem is at the heart of many data mining problems [21, 13].

The rewritings computation in the hypergraph framework can be formulated as follows: Let $\mathcal{H}_B = (V_B, E_B)$ be a hypergraph. Let Q be a query such that $Q \equiv \sqcap_{i=1}^{n} \forall w_i.P_i$. Each view or conjunction of views occurring in the buckets $B(w_i, P_i)$ is associated to a vertex in V_B. The set E_B consists of the set of the buckets $B(w_i, P_i)$ themselves.

Example 4. Let $Q \equiv \forall w_1.A_1 \sqcap \forall w_2.A_2 \sqcap \forall w_3.A_3 \sqcap \forall w_4.A_4$
and the associated buckets:

$B(w_1, A_1)$	$B(w_2, A_2)$	$B(w_3, A_3)$	$B(w_4, A_4)$
V_1	V_1	V_3	$V_3 \sqcap V_4$
V_2	V_4	V_4	$V_1 \sqcap V_2 \sqcap V_3$
	$V_3 \sqcap V_4$		

Let V_{34} be a representation of $V_3 \sqcap V_4$ and V_{123} of $V_1 \sqcap V_2 \sqcap V_3$. A hypergraph $\mathcal{H}_B$ can be built over $V_B = \{V_1, V_2, V_3, V_4, V_{34}, V_{123}\}$ as follows: $E_B = \{\{V_1, V_2\}, \{V_1, V_4, V_{34}\}, \{V_3, V_4\}, \{V_{34}, V_{123}\}\}$

The following theorem shows that the minimal transversals of this hypergraph are a superset of the maximally-contained rewritings of the query Q.

Theorem 3. *Let Q be a query and its buckets $B(w_i, P_i)$, with $i \in [1, n]$. Let $\mathcal{H}_B = (V_B, E_B)$ be a hypergraph defined in terms of the $B(w_i, P_i)$. Let $T_{\mathcal{H}_B}$ be the set of minimal transversals of $\mathcal{H}_B$.*

Then $\mathcal{MCR} \subseteq T_{\mathcal{H}_B}$.

Proof of this theorem is given in annex C.2, page 22.

Then, as in conventional bucket-like approaches, the generation of the query rewritings, here the minimal transversals computation of $\mathcal{H}_B$, requires the deletion of inconsistent and non maximal rewritings. The following example illustrates the maximally-contained rewritings computation of a given query Q from the hypergraph $\mathcal{H}_B$ obtained in example 4.

Example 5. The minimal transversals of $\mathcal{H}_B$ given in example 4 are: $\{V_1, V_3, V_{34}\}$, $\{V_1, V_3, V_{123}\}$, $\{V_1, V_4, V_{34}\}$, $\{V_1, V_4, V_{123}\}$, $\{V_2, V_4, V_{34}\}$, $\{V_2, V_4, V_{123}\}$, $\{V_2, V_3, V_{34}\}$, $\{V_2, V_3, V_{123}\}$.

After expansion of the minimal transversals, we obtain the following set of candidate maximally-contained rewritings: $\{V_1 \sqcap V_3 \sqcap V_4, V_1 \sqcap V_2 \sqcap V_3, V_1 \sqcap V_2 \sqcap V_3 \sqcap V_4, V_2 \sqcap V_3 \sqcap V_4\}$.

Some of them are not minimal. The final set of maximally-contained rewritings is then: $\{V_1 \sqcap V_3 \sqcap V_4, V_1 \sqcap V_2 \sqcap V_3, V_2 \sqcap V_3 \sqcap V_4\}$.

This approach reduces significantly the number of candidate rewritings in comparison with the cartesian product. In example 4, 8 candidates are generated instead of 24 using the cartesian product.

The efficiency and scalability of our query rewriting algorithm *ComputeRew* depends on the computation of $RewS_1(E, w)$ and $RewS_2(n, w.R_v)$ since their number of candidates is exponential in the number of views. All other cases involve only one view and therefore are not concerned by scalability. The max-rewritings generation can also be costly even with the hypergraph-based characterization, due to the potentially large number of elements to generate. To cope with these difficulties, our work features the use of data mining techniques to devise an efficient algorithm that favor scalability w.r.t. the number of views in both steps. To do that, we use a data mining library called *iZi*.

5 Query rewriting alorithm in $\mathcal{ALN}(\mathcal{O}_v)$using *iZi*

iZi [11] is a generic C++ library for pattern mining problems known to be "representable as sets", i.e., those problems whose solution space is isomorphic to a boolean lattice. The basic idea of *iZi* is to offer a toolbox for a rapid and easy development of efficient and robust data mining programs. This library takes advantage of a well established theoretical framework from an implementation point of view by providing efficient data structures for boolean lattice representation and several implementations of well known algorithms. By the way, these problems can be implemented with only minimal effort, i.e., programmers do not have to be aware of low-level code, customized data structures and algorithms being available for free. This library has been devised and applied to several problems such as itemset mining and constraint mining in relational databases.

Following the guidelines given with *iZi*, the rest of this section shows how three subparts of the query rewriting algorithm can take advantage of *iZi*. First we recall the underlying theoretical framework and then we point out in details how *iZi* can be used in this context.

5.1 A theoretical framework for Knowledge Discovery

We recall in this section the theoretical KDD framework defined in [21] for interesting pattern discovery problems, and used in *iZi*. Such a framework has been successfully applied in different contexts such as association rules [3], functional dependencies [16] and inclusion dependencies [9] to mention a few.

Given a database r, a finite language $\mathcal{L}$ for expressing patterns or defining subgroups of the data, and a predicate P for evaluating whether a pattern $\varphi \in \mathcal{L}$ is true or "interesting" in r, the discovery task is to find the theory of r with respect to $\mathcal{L}$ and P, i.e., the set $Th(r, \mathcal{L}, P) = \{\varphi \in \mathcal{L} \mid P(r, \varphi) \text{ is true}\}$.

Let us suppose a specialization/generalization relation between patterns of $\mathcal{L}$. Such a relation is a partial order $\preceq$ on the patterns of $\mathcal{L}$. We say that φ is more general (resp. more specific) than θ, if $\varphi \preceq \theta$ (resp. $\theta \preceq \varphi$).

Let $(I, \preceq)$ be a partially ordered set of elements. A set $S \subseteq I$ is *closed downwards* (resp. *closed upwards*) if, for all $X \in S$, all subsets (resp. supersets) of X are also in S.

The predicate P is said to be monotone (resp. anti-monotone) with respect to $\preceq$ if for all $\theta, \varphi \in \mathcal{L}$ such that $\varphi \preceq \theta$, if $P(r, \varphi)$ is true (resp. false) then $P(r, \theta)$ is true (resp. false). As a consequence, if the predicate is monotone (resp. anti-monotone), the set $Th(r, \mathcal{L}, P)$ is upward (resp. downward) closed, and can be represented by either of the following sets:

- its *positive border*, denoted by $\mathcal{B}d^+(\ Th(r, \mathcal{L}, P)\ )$, made up of the MOST SPECIALIZED true patterns when $Th(r, \mathcal{L}, P)$ is downward closed, and the MOST SPECIALIZED false patterns when $Th(r, \mathcal{L}, P)$ is upward closed;
- its *negative border*, denoted by $\mathcal{B}d^-(\ Th(r, \mathcal{L}, P)\ )$, made up of the MOST GENERALIZED false patterns when $Th(r, \mathcal{L}, P)$ is downward closed, and the MOST GENERALIZED true patterns when $Th(r, \mathcal{L}, P)$ is upward closed.

The union of these two borders is called the *border of* $Th(r, \mathcal{L}, P)$, and is denoted by $\mathcal{B}d(\ Th(r, \mathcal{L}, P)\ )$.

The last hypothesis of this framework is that the problem must be representable as sets via an isomorphism, i.e., the search space can be represented by a boolean lattice (or subset lattice). Let $(\mathcal{L}, \preceq)$ be the ordered set of all the patterns defined by the language $\mathcal{L}$. Let C be a finite set of elements. The problem is said to be *representable as sets* if a bijective function $f : (\mathcal{L}, \preceq) \rightarrow (2^C, \subseteq)$ exists and its inverse function f^{-1} is computable, such that:

$$X \preceq Y \Longleftrightarrow f(X) \subseteq f(Y)$$

In the sequel, a problem representable as sets will be referred to as "isomorphic to a boolean lattice".

A salient feature of this latter restriction relies on the notion of dualization [21, 13], well known in combinatorics as minimal transversals of a hypergraph.

5.2 Three scalable components of the query rewriting algorithm

In our query rewriting algorithm, three enumeration problems have been identified as possible bottleneck: $S_1(E, w)$ computation, $S_2(n, w.R_v)$ computation and minimal transversal of a hypergraph. This section shows how these three problems can be reformulated in this framework.

Reformulating the problems as pattern mining problems Problems of the framework have to be *enumeration problems under constraints*, i.e., of the form "enumerate all the patterns that satisfy a condition". Consequently, the first step is to reformulate our problems in such pattern mining problems.

$S_1(E, w)$ **subproblem:** $S_1(E, w)$ consists in extracting the maximally-contained rewritings of the conjunct $\forall w.E$ of Q. In other words, the problem is to enumerate all minimal subsets of $\mathcal{V}$ whose intersection of their restricted set of values for the word w is included in E.
$S_1(E, w) = min_{\subseteq}(U \in 2^{\mathcal{V}} | \forall V_i \in U, \forall w.E_i \in V_i \text{ and } \bigcap_{V_i \in U} E_i \subseteq E)$

$S_2(n, w.R_v)$ **subproblem:** $S_2(n, w.R_v)$ consists in extracting the maximally-contained rewritings of the conjunct $\forall w. \leq n\ R_v$ of Q. In other words, the problem is to enumerate all minimal subsets of $\mathcal{V}$ whose cardinality of the intersection of their restricted set of values for the word $w.R_v$ is smaller or equal to n.
$S_2(n, w.R_v) = min_{\subseteq}(U \in 2^{\mathcal{V}} | \forall V_i \in U, \forall w.R_v.E_i \in V_i \text{ and } |\bigcap_{V_i \in U} E_i| \leq n)$

Max-rewritings generation (from Section 4.2) : Let $Q \equiv \sqcap_{i=1}^{n} \forall w_i.P_i$ be a query and B its buckets, i.e., $B = \bigcup_{i=1}^{n} B(w_i, P_i)$. Thanks to the hypergraph-based characterization, the maximal-contained rewritings, or $\mathcal{MCR}$, generation problem can be reformulated as enumerating all minimal transversals of the hypergraph $\mathcal{H}_B = (V_B, E_B)$, where V_B is composed of all the views or conjunction of views of the buckets ($V_B = \{v \mid v \in B(w_i, P_i), \forall i \in [1, n]\}$) and E_B consists of the set associated with each bucket ($E_B = \{e \mid e \in B\}$) .
$\mathcal{T}_{\mathcal{H}_B} = \{X \subseteq V_B \mid X \text{ minimal traversal of } \mathcal{H}_B\}$

Defining the language, the predicate and proving monotonicity Once a problem is suspected to fit into the framework, the pattern language, the predicate and the partial order among patterns must be properly defined to go further. Moreover, predicate monotonicity has to be proven. In this subsection, we check all these aspects to fit our three subproblems in the theoretical framework. Proofs of properties and theorems of this subsection are given in annex D, page 22.

$S_1(E, w)$ **subproblem:**

1. The pattern language is $\mathcal{L}_{S_1(E,w)} = \{X \mid X \subseteq \mathcal{V}\}$
2. The predicate $P_{S_1(E,w)}(E, X)$ is true iff for all $V_i \in X$ and $w.E_i \in V_i$, $\bigcap_{V_i \in X} E_i \not\subseteq E$.
3. The partial order over $\mathcal{L}_{S_1(E,w)}$ is the set inclusion $\subseteq$.

Let X be a set of views satisfying $P_{S_1(E,w)}$, i.e., $P_{S_1(E,w)}(E, X) = \text{true}$. It is clear that any subset Y of X also satisfies $P_{S_1(E,w)}$, since $\bigcap_{V_i \in X} E_i \subseteq \bigcap_{V_j \in Y} E_j$.

Property 1. The predicate $P_{S_1(E,w)}(E, X)$ is anti-monotone w.r.t. set inclusion.

The $S_1(E, w)$ problem can be reformulated as follows:

Theorem 4. $S_1(E, w) = \mathcal{B}d^-(Th(E, \mathcal{L}_{S_1(E,w)}, P_{S_1(E,w)}))$

$S_2(n, w.R_v)$ **subproblem:**

1. The pattern language is $\mathcal{L}_{S_2(n,w.R_v)} = \{X \mid X \subseteq \mathcal{V}\}$
2. The predicate $P_{S_2(n,w.R_v)}(n, X)$ is true iff for all $V_i \in X$ and $w.R_v.E_i \in V_i$, $|\bigcap_{V_i \in X} E_i| > n$.
3. The partial order is $\subseteq$.

Let X be a set of views satisfying $P_{S_2(n,w.R_v)}$, i.e., $P_{S_2(n,w.R_v)}(n, X) = \text{true}$. It is clear that any subset Y of X also satisfies $P_{S_2(n,w.R_v)}$, since $|\bigcap_{V_i \in X} E_i| \leq |\bigcap_{V_j \in Y} E_j|$.

Property 2. The predicate $P_{S_2(n,w.R_v)}(n, X)$ is anti-monotone w.r.t. set inclusion.

The $S_2(n, w.R_v)$ problem can be reformulated as follows:

Theorem 5. $S_2(n, w.R_v) = \mathcal{B}d^-(Th(n, \mathcal{L}_{S_2(n,w.R_v)}, P_{S_2(n,w.R_v)}))$

$\mathcal{T}_{\mathcal{H}_B}$ **subproblem:**

1. The pattern language is $\mathcal{L}_{\mathcal{T}_{\mathcal{H}_B}} = \{X \mid X \subseteq V_B\}$
2. The predicate $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, X)$ is true iff X is not a transversal of $\mathcal{H}_B$, i.e., if $\exists H \in E_B$ such as $X \cap H = \emptyset$.
3. The partial order is $\subseteq$.

It is also clear that any subset of non-transversal element is also non-transversal.

Property 3. The predicate $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, X)$ is anti-monotone w.r.t. set inclusion.

The minimal transversals generation problem can be reformulated as follows:

Theorem 6. $\mathcal{T}_{\mathcal{H}_B} = \mathcal{B}d^-(Th(\mathcal{H}_B, \mathcal{L}_{MaxRew(B)}, P_{MaxRew(B)}))$

Clearly, our problems are representable as sets, i.e., isomorphic to a boolean lattice. The function f given in Section 5.1 is the identity function. Consequently, the iZi library can be directly used to solve these three subproblems.

6 Experimental evaluation

A query rewriting prototype (figure 1) has been implemented based on the theoretical investigations introduced so far. It takes as input a query Q expressed in terms of schema $\mathcal{S}$ and returns the set of all the maximally-contained rewritings of Q. The prototype is composed of two parts. The first one parses and normalizes the query Q from the schema $\mathcal{S}$ stored in a database. The second one, i.e., `ComputeRew`, is devoted to the computation of the query rewritings. This part consists of two components: `BucketsComputing` and `RewritingGeneration`. As shown by the algorithm 2, the `BucketsComputing` component requires as input the views $\mathcal{V}$ stored in a database. Moreover, as seen in Section 5, both components use the `iZi` library.

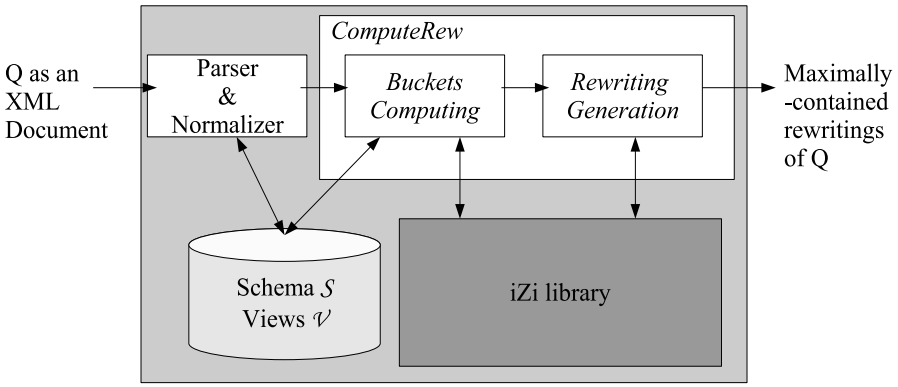

Fig. 1. Prototype description

`BucketsComputing` and `RewritingGeneration` (the most costly operations of our prototype) have been implemented using the generic APriori-like implementation provided in `iZi` [11]. The use of the APriori-like algorithm is motivated by two main reasons. First, it gives without any overhead the negative border, i.e., the solution of our subproblems. Second, several benchmarks [6, 5] have shown that this algorithm is particularly efficient for discovering "not too large" solutions, which turns out to be the case in our experiments (see below).

Our implementation has been evaluated on synthetic dataset. Our objective has been to show the scalability of our proposition with respect to the number of views. More particularly, we

focus on the three most costly steps of our implementation. Our first experiments focus on the computation of the sets S_1 and S_2, i.e., the computation of the rewritings due to value constraints. Second, we experiment the minimal traversals computation. The experimentations were performed on a PC with 2.6GHz P4 pro CPU and 3Go RAM.

S_1 and S_2 computation In this first part of the experimentations, synthetic datasets have the following characteristics. The values of constraints are chosen among 33000. In figure 2, cardinality of the constraints is less than 10 while the number of views takes its values in the set $\{3000, 5000, 10000, 15000\}$.

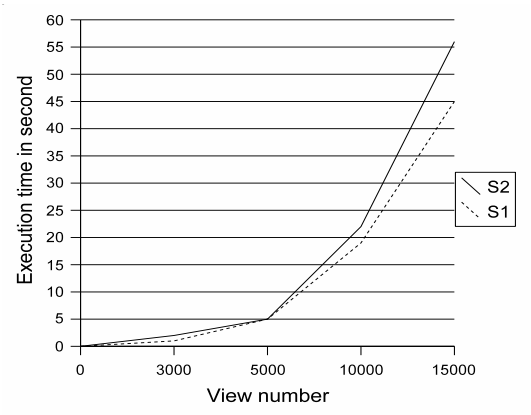

Fig. 2. Performances of S_1 and S_2 computation

In figure 3, the number of views is fixed to 5000 while the maximal cardinality of the constraints is either 10 or 20 or 30 or 40.

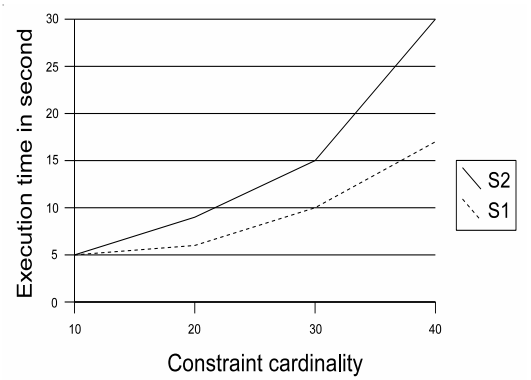

Fig. 3. Effect of value constraints cardinalities on performance

Figure 2 shows that our implementation handles up to 15000 views in an acceptable time, less than 60 seconds. Until about 10000 views, implementation remains efficient. In figure 3, we fix the number of views to 5000 and concentrate on the impact of constraints cardinality on the execution time. The implementation is very efficient for value constraints having a cardinality less than 30.

Minimal transversal computation In our context, one of the problem for the minimal transversal computation is the huge number of vertices (i.e., views) that may occur in the same edge (i.e., bucket). To reduce this number, an optimization has been brought to the minimal transversal computation. Actually, the idea is to drastically reduce the number of vertices by regrouping together vertices which belong to the same edges. For example, if vertices a and b belong to the same edges, these two vertices can be replaced by a unique vertice a'. Then, the minimal transversal computation is applied on this "reduced" hypergraph. At the end, to have the solutions of the initial hypergraph, each transversal containing a' is replaced by two transversals: one with a instead of a' and one with b instead of a'. Thanks to the characteristics of our hypergraphs, i.e., a very small number of edges and huge number of vertices, this optimization is very effective in practice.

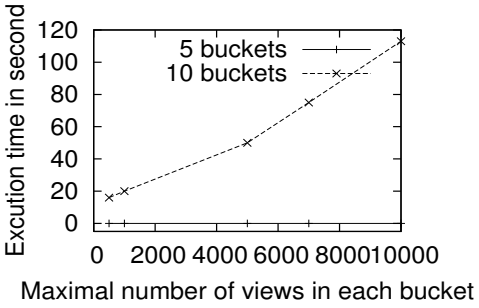

Fig. 4. Performance of the minimal transversal computation

The datasets used in the experiments are characterized by their number of buckets (i.e., number of edges of the hypergraph), their maximal number of views or conjunction of views in the buckets (i.e., the maximal size of the edges) and their total number of views or conjunction of views (i.e., the total number of vertices). The figure 4 presents the execution time for datasets with 5 and 10 buckets. The maximal number of views (or conjunction of views), in x-axis, is equal to the total number of views. As shown by this figure, our implementation can handle 10000 views instantaneously when processing 5 buckets. For the dataset with 10 buckets, it can process until 10000 views in an acceptable time. Even if this number of buckets seems small, recall that each bucket corresponds to a condition in the initial query, and consequently having more than 10 conditions for a single query stills rare.

For more buckets, despite the use of scalable data structures in the implementation, the cost of rewriting generation remains high. However, such implementation improves significantly an approach that would compute a cartesian product. In particular, our optimization for the minimal transversal computation reduced the number of vertices by a factor of 1.5 to 20 according to the datasets being studied. Moreover, even if the worst case (i.e., the cartesian product) can hardly be optimized, this case remains rare in our application since we have lot of views and a small number of buckets. On the other hand, our application supports the creation of neighborhood vertices. Consequently, our query rewriting prototype can take advantage of the two optimizations: transversal minimal computation and neighborhood vertices aggregation.

Experimental results show clearly the feasibility and scalability of our approach.

7 Conclusion

Our work supplies innovative and complementary contribution to existing works on answering queries using views by considering a new kind of constraints that can be very useful in practical situations. More precisely, we investigated this problem in the setting of the logic $\mathcal{ALN}(\mathcal{O}_v)$

that allows the expression of value constraints. We show that it is possible to compute all the certain answers of a given query Q by computing its maximally-contained rewritings. Furthermore, our work is the first to use efficient data mining techniques [11] to improve the scalability of a query rewriting Bucket-like algorithm. A query rewriting prototype has been implemented. This prototype is based on an existing data mining tool [11] for the bucket construction and for the global rewritings computation. Experimental results confirm the interest of our approach.

References

1. Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *PODS'98, Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 254–263. ACM Press, 1998.
2. Foto N. Afrati, Chen Li, and Prasenjit Mitra. Answering queries using views with arithmetic comparisons. In Lucian Popa, editor, *PODS'02, Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 209–220. ACM, 2002.
3. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of the International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.
4. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
5. Roberto J. Bayardo Jr., Bart Goethals, and Mohammed Javeed Zaki, editors. *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, UK, November*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
6. Roberto J. Bayardo Jr. and Mohammed Javeed Zaki, editors. *FIMI '03, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, USA, November*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.
7. C. Beeri, A. Halevy, and M.C. Rousset. Rewriting Queries Using Views in Description Logics. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *PODS'97, Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems , New York, NY, May 12–14, Tucson, Arizona*, pages 99–108. ACM Press, 1997.
8. Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research (JAIR)*, 1:277–308, 1994.
9. Fabien De Marchi and Jean-Marc Petit. Zigzag: a new algorithm for mining large inclusion dependencies in database. In *ICDM'03, Proceedings of the IEEE International Conference on Data Mining*, pages 27–34. IEEE Computer Society, 2003.
10. Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, 1995.
11. Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit. iZi: a new toolkit for pattern mining problems. In Aijun An, Nick Cercone, Stan Matwin, Zbigniew W. Ras, and Dominik Slezak, editors, *ISMIS'08, International Symposium on Methodologies for Intelligent Systems*, Lecture Notes in Computer Science, pages 1–6. Springer, 2008.
12. François Goasdoué and Marie-Christine Rousset. Answering queries using views: A krdb perspective for the semantic web. *ACM Transactions on Internet Technology*, 4(3):255–288, 2004.
13. Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. Discovering all most specific sentences. *ACM transactions on database systems*, 28(2):140–174, 2003.
14. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
15. Ian Horrocks and Ulrike Sattler. Ontology reasoning in the shoq(d) description logic. In Bernhard Nebel, editor, *IJCAI'01, International Joint Conferences on Artificial Intelligence*, pages 199–204. Morgan Kaufmann, 2001.
16. Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Computer Journal*, 42(2):100–111, 1999.
17. Leonid Khachiyan, Endre Boros, Khaled M. Elbassioni, and Vladimir Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discrete Applied Mathematics*, 154(16):2350–2372, 2006.
18. Ralf Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Computer Science*. Springer, 2001.

19. M. Lenzerini. Data integration : A theoretical perspective. In *PODS'02, Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Madison, Wisconsin, 2002.
20. Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *VLDB'96, Proceedings of the International Conference on Very Large Data Bases*, pages 251–262. Morgan Kaufmann, 1996.
21. Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data mining and knowledge discovery*, 1(3):241–258, 1997.
22. Rachel Pottinger and Alon Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB Journal*, 10(2-3):182–198, 2001.
23. Andrea Schaerf. Reasoning with individuals in concept languages. *Data & Knowledge Engineering*, 13(2):141–176, 1994.
24. Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

A Subsumption characterization

Normalization rules that allow to transform $\mathcal{ALN}(\mathcal{O}_v)$ concepts into their normal forms are described in the table 2. Letter A denotes an atomic concept. Letters E, E_1 and E_2 denote set of values while D and D' specify any kind of concept. To simplify the set of normalization rules, we assume that any description like $(\geq 0R)$ or $(\forall R_C.\top_C)$ or $(\forall R_V.\top_V)$ is transformed into $\top_C$ while $D \sqcap \top_C$ (if D is not a set of values otherwise $D \sqcap \top_C$ is inconsistent) and $E \sqcap \top_V$ become respectively D and E. First rules (1) and (2) are recursively applied until a saturation point. Next

(1) $\forall w.D \sqcap \forall w.D' \rightarrow \forall w.(D \sqcap D')$
(2) $E_1 \sqcap E_2 \rightarrow E$ such that $E = E_1 \cap E_2$
(3) $\forall R_v.E \rightarrow \forall R_v.E \sqcap (\leq kR_v)$, where $\lvert E\rvert = k$
(4) $\leq 0R \sqcap \forall R.D \rightarrow \leq 0R$
(5) $A \sqcap \neg A \rightarrow \bot$
(6) $(\geq nR) \sqcap (\leq mR) \rightarrow \bot$ if $n > m$
(7) $(\geq nR) \sqcap (\geq mR) \rightarrow (\geq max(n,m)R)$
(8) $(\leq nR) \sqcap (\leq mR) \rightarrow (\leq min(n,m)R)$
(9) $D \sqcap \bot \rightarrow \bot$
(10) $\forall R.\bot \rightarrow \leq 0R$
(11) $\forall w.(D \sqcap D') \rightarrow \forall w.D \sqcap \forall w.D'$

Table 2. Normalization rules.

the rule (3) is applied only once. Rules (4) to (10) are then applied recursively until a saturation point. Finally, the rule (11) is applied recursively until a saturation point.

We present now the proof of the theorem 1.

Proof (Proof of theorem 1).

- completeness ($\Rightarrow$) The proof of the completeness of this theorem is derived from the structural characterization of subsumption in Classic logics [8].
  Let C and D be two concept descriptions. Assume that D is in its normal form, i.e., $D \equiv \forall w_1.P_1 \sqcap ... \sqcap \forall w_n.P_n$. Let G_C be the canonical description graph associated with C and assume that the subsumption algorithm given in [8] returns true with the input D and G_C, thas is, $C \sqsubseteq D$. Therefore, G_C verifies the conditions stated in [8].
  We can construct a concept from the graph G_C, and then apply rule 11 of table 2 to expand it so there are no nested conjunctions. We get then a description of C in our expected normal form that verifies the following conditions:

either $C \equiv \bot$ or $D \equiv \top_C$, or
for every $\forall w.P \in D$, we have
(a) $\forall w.P \in C$ or,
(b) $\forall w.E' \in C$ with $E' \subseteq E$ if $P = E$ or,
(c) $\forall w.(\leq kR) \in C$ with $k \leq n$ if $P = \leq nR$ or,
(d) $\forall w.R.E \in C$ with $|E| \leq n$ if $P = \leq nR$ and $R \in \mathcal{R}_v$ or,
(e) $\forall w.(\geq kR) \in C$ with $k \geq n$ if $P = \geq nR$ or,
(f) $\forall v.(\leq 0R) \in C$ with vR prefix of w.
These conditions are those stated by theorem 1.

– soundness ($\Leftarrow$) Let C and D be two concept descriptions in their normal form such that $D \equiv \sqcap_{i=1}^{n} \forall w_i.P_i$. We have that either condition 1) or conditions 2) of theorem 1 are verified for each $\forall w_i.P_i$ in D. We want to show that implies $C \sqsubseteq D$.
  1) if $C \equiv \bot$ then the interpretation of C is empty and any description D subsumes it. if $D \equiv \top_C$ then its interpretation is the set of all *classic* individuals, i.e. δ_C. Since any concept in $\mathcal{ALN}(\mathcal{O}_v)$ is a subset of δ_C any concept C is subsumed by $\top_C$.
  2) otherwise one of the following condition occurs
     * if $\forall v_i.\bot$ where $w_i = v_i u$ belongs to C. Hence $C \sqsubseteq \forall v_i.\bot \sqsubseteq \forall v_i.u.P_i$ and $C \sqsubseteq \forall w_i.P_i$.
     * if $P_i = A$ or $P_i = \neg A$ then $\forall w_i.P_i$ is in the description of C and $C \sqsubseteq \forall w_i.P_i$.
     * if $P_i = (\leq nR)$ then $\forall w_i.(\leq kR)$ where $k \leq n$ is in the description of C and $C \sqsubseteq \forall w_i.(\leq kR) \sqsubseteq \forall w_i.P_i$, because $k \leq n$.
     * if $P_i = (\geq nR)$ then $\forall w_i.(\geq kR)$ where $k \geq n$ is in the description of C and $C \sqsubseteq \forall w_i.(\geq kR) \sqsubseteq \forall w_i.P_i$, because $k \geq n$.
     * if $P_i = E$ then $\forall w_i.E'$ where $E' \subseteq E$ is in the description of C and $C \sqsubseteq \forall w_i.E' \sqsubseteq \forall w_i.P_i$, because $E' \subseteq E$.

     Hence for all $\forall w_i.P_i$ in D we have $C \sqsubseteq \forall w_i.P_i$ and $C \sqsubseteq \sqcap_{i=1}^{n} \forall w_i.P_i$ and $C \sqsubseteq D$

B From query answering using views to query rewriting using views

Let us given the proof of lemma 1

Proof (Proof of lemma 1).
Let t be a certain answer.
Let $Q \equiv \forall w_1.P_1 \sqcap \ldots \sqcap \forall w_n.P_n$ a query.
Let $\mathcal{I}$ be a model of $\mathcal{S}$, s.t. $V^{ext} \subseteq V^{\mathcal{I}}$, $\forall\, V \in \mathcal{V}$.
We want to show that if t is a certain answer of Q, i.e., $t \in Q^{\mathcal{I}}$, then there exists a conjunction C of views from $\mathcal{V}$ s.t. $C \sqsubseteq Q$ and $t \in C^{ext}$.

According to the definition 1 on certain answers, if t is a certain answer of Q, then $t \in \mathcal{V}^{ext}$. There exists then a subset of $\mathcal{V}$ whose the views contains t in their extension.
Let M be the set of all the views that contains t in their extension. Let C_M, the conjunction of views in M. Therefore $t \in C_M^{ext}$.

It remains to show that C_M is necessarily subsumed by the query Q. To achieve that, we assume that C_M is not subsumed by Q and we show that it is possible to find an interpretation $\mathcal{J}$, model of $(\mathcal{S}, \mathcal{V})$ in which t does not belong to $Q^{\mathcal{J}}$. Therefore t is not a certain answer.

We want to show that $C_M \sqsubseteq Q$, i.e., that $\forall \mathcal{I}$, model of $(\mathcal{S}, \mathcal{V})$ s.t. $V^{ext} \subseteq V^{\mathcal{I}}$ for each view $V \in \mathcal{V}$, we have $C_M^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$.
Assume that $C_M \not\sqsubseteq Q$, then according to theorem 1, there exists $\forall w_k.P_k \in Q$ s.t. $C_M \not\sqsubseteq \forall w_k.P_k$. Consequently, there exists an interpretation $\mathcal{K}$, model of $(\mathcal{S}, \mathcal{V})$ with $V^{ext} \subseteq V^{\mathcal{K}}$ for each view $V \in \mathcal{V}$, s.t. $C_M^{\mathcal{K}} \not\subseteq \forall w_k.P_k^{\mathcal{K}}$. Therefore there exists x in $\delta_C^{\mathcal{K}}$ s.t. $x \in C_M^{\mathcal{K}}$ but s.t. $x \notin (\forall w_k.P_k)^{\mathcal{K}}$.
If x was t, then t could not be a certain answer. We are then going to define an interpretation $\mathcal{J}$, model of $(\mathcal{S}, \mathcal{V})$ with $V^{ext} \subseteq V^{\mathcal{J}}$ for each view $V \in \mathcal{V}$, s.t. there exists $x = t$ in $\delta_C^{\mathcal{J}}$ with $t \in C_M^{\mathcal{J}}$ but s.t. $t \notin \forall w_k.P_k^{\mathcal{J}}$.

Let $\mathcal{J}$ an interpretation defined as follow: **(i)** $\forall i \in [1,n]$, we have $(\forall w_i.P_i)^{\mathcal{J}} = (\forall w_i.P_i)^{\mathcal{K}}$, **(ii)** $(\forall w_k.P_k)^{\mathcal{J}} = (\forall w_i.P_i)^{\mathcal{K}} - \{t\} \cup \{t'\}$ with t' a new individual ($\Delta^{\mathcal{J}} = \Delta^{\mathcal{K}} \cup \{t'\}$) and **(iii)** every $(x,t) \in R^{\mathcal{K}}$ is replaced by $(x,t') \in R^{\mathcal{J}}$ and respectively, every $(t,y) \in R^{\mathcal{K}}$ is replaced by $(t',y) \in R^{\mathcal{J}}$.

Show now that $\mathcal{J}$ remains consistent with the view extensions, i.e., $V^{ext} \subseteq V^{\mathcal{J}}$ for each view $V \in \mathcal{V}$.

For the views in M:

The views V in M are not subsumed by $\forall w_k.P_k$ and are s.t. $t \in V^{\mathcal{K}}$. Therefore, replacing t by t' in $\forall w_k.P_k^{\mathcal{J}}$ has no impact over the interpretation of $V^{\mathcal{J}}$ Moreover, the transformation of $\mathcal{K}$ in $\mathcal{J}$ preserves the semantics of every axioms in $\mathcal{V}$ thanks to the property **(iii)** that defines $\mathcal{J}$.

Therefore, for each view V in M, we have $V^{\mathcal{J}} = V^{\mathcal{K}}$ and the relationship $V^{ext} \subseteq V^{\mathcal{J}}$ is verified.

For the views in $\mathcal{V} \backslash M$:

The views V in $\mathcal{V} \backslash M$ are s.t. $t \notin V^{\mathcal{K}}$. Consequently, the deletion of t in $\forall w_k.P_k^{\mathcal{J}}$ does not modify the interpretation $V^{\mathcal{J}}$. Moreover, according to the property **(iii)** that defines $\mathcal{J}$, for each view V in $\mathcal{V} \backslash M$, we have $V^{\mathcal{J}} = V^{\mathcal{K}}$ and the relationship $V^{ext} \subseteq V^{\mathcal{J}}$ is verified.

Then there exists an interpretation $\mathcal{J}$ of $(\mathcal{S}, \mathcal{V})$, s.t. $V^{ext} \subseteq V^{\mathcal{J}}$, $\forall V \in \mathcal{V}$, in which t does not belong to $Q^{\mathcal{J}}$. Consequently, t is not a certain answer of Q which contradicts the initial assumption. Therefore we have $C_M \sqsubseteq Q$.

Now follows proof of theorem 2.

Proof (Proof of theorem 2). The proof of this theorem lies on the following principle. Each Q_i is a maximally-contained rewriting of Q. Thus by definition, Q_i is subsumed by Q. Consequently, we have $Q_i(\mathcal{V}^{ext}) \subseteq Ans(Q, \mathcal{V}^{ext})$ for every $i \in \{1, \ldots, n\}$ and $\cup_{i=1}^{n} Q_i(\mathcal{V}^{ext}) \subseteq Ans(Q, \mathcal{V}^{ext})$. It remains to show that the set of certain answers is contained in $\cup_{i=1}^{n} Q_i^{ext}$. Let t be a certain answer, then there exists a conjunction Q' of views s.t. $t \in Q'(\mathcal{V}^{ext})$ and $Q' \sqsubseteq Q$. Therefore either Q' is maximally-contained in Q and there exists $i \in \{1, \ldots, n\}$ s.t. $Q' \equiv Q_i$, or there exists $Q_i \in \{Q_1, \ldots, Q_n\}$ s.t. $Q' \sqsubseteq Q_i$. Consequently, we have $t \in Q_i(\mathcal{V}^{ext})$.

The proof of lemma 2 lies on lemma 4 below that characterizes the subsumption between two consistent conjunctions of views.

Lemma 4. *Let $\mathcal{V}$ be a terminology in $\mathcal{ALN}(\mathcal{O}_v)$. Let Q_1 and Q_2 two consistent conjunctions of views in $\mathcal{V}$.*

$Q_1 \sqsubseteq Q_2$ iff Q_2 is made of a subset of views occurring in Q_1.

Proof (Proof of lemma 4).

($\Leftarrow$) Let Q_{i_1} and Q_{i_2} be two conjunctions of views in $\mathcal{V}$. We are going to show that if Q_{i_2} refers a subset of views in Q_{i_1} then $Q_{i_1} \sqsubseteq Q_{i_2}$.
Assume that $Q_{i_1} \equiv V_1 \sqcap \ldots \sqcap V_n$.
Q_{i_2} refers only a subset of $\{V_1, \ldots, V_n\}$.
Assume that Q_{i_2} is equivalent to the following expression: $Q_{i_2} \equiv V_1 \sqcap \ldots \sqcap V_{j-1} \sqcap V_{j+1} \sqcap \ldots \sqcap V_n$.
Therefore, since the used formal framework is $\mathcal{ALN}(\mathcal{O}_v)$, for all interpretation $\mathcal{I}$, we have $Q_{i_1}^{\mathcal{I}} \subseteq Q_{i_2}^{\mathcal{I}}$.

($\Rightarrow$) Let Q_{i_1} and Q_{i_2} be two consistent conjunctions of views from $\mathcal{V}$. We are going to show that if $Q_{i_1} \sqsubseteq Q_{i_2}$ then Q_{i_2} refers a subset of views occurring in Q_{i_1}.
Since $\mathcal{V}$ is a primitive terminology that is normalized and expanded, each view V_{i_j} is a conjunction between a concept description D_{i_j} and a unique atomic concept $\overline{V_{i_j}}$.
Assume that $Q_{i_1} \equiv V_1 \sqcap \ldots \sqcap V_n$, where $V_i \in \mathcal{V}$ for all $i \in \{1, ..., n\}$
then $Q_{i_1} \equiv (D_1 \sqcap \ldots \sqcap D_n) \sqcap (\overline{V_1} \sqcap \ldots \sqcap \overline{V_n})$.
Every concept that subsumes Q_{i_1} must contain in its description a conjunction of views from the set $\{\overline{V_1}, \ldots, \overline{V_n}\}$.
In other words, every concept that subsumes Q_{i_1} refers a subset of $\{\overline{V_1}, \ldots, \overline{V_n}\}$.
Since every concept $\overline{V_i}$ with $i \in \{1, ..., n\}$, is an unique atomic concept associated with a single view V_i, every concept that subsumes Q_{i_1} refers a subset of $\{V_1, \ldots, V_n\}$.

Proof of lemma 2 is given below.

Proof (Proof of lemma 2).

($\Rightarrow$) Assume that $Q' \equiv V_1 \sqcap \ldots \sqcap V_n$ is a maximally-contained and conjunctive rewriting Q in terms of $\mathcal{V}$.
We want to show that for every concept Q'' obtained by removing from Q' one of its conjuncts V_i, we have $Q'' \not\sqsubseteq Q$.
Assume that $Q'' \equiv V_1 \sqcap ... \sqcap V_{j-1} \sqcap V_{j+1} \sqcap ... \sqcap V_n$, obtained by removing from Q' one view V_j, with $j \in \{1, ..., n\}$, is subsumed by Q (i.e. $Q'' \sqsubseteq Q$). We have also $Q' \sqsubseteq Q''$. We are going to show that in this case, Q' is not maximally contained in Q.
To achieve that, we show that Q'' is not equivalent to Q'.
Let $Q' \equiv V_1 \sqcap ... \sqcap V_n$ and $Q'' \equiv V_1 \sqcap ... \sqcap V_{j-1} \sqcap V_{j+1} \sqcap ... \sqcap V_n$. According to the open word assumption,
$Q'' \equiv (D_1 \sqcap \ldots \sqcap D_{(j-1)} \sqcap D_{(j+1)} \sqcap \ldots \sqcap D_n) \sqcap (A_1 \sqcap \ldots \sqcap A_{(j-1)} \sqcap A_{(j+1)} \sqcap \ldots \sqcap A_n)$
$Q' \equiv (D_1 \sqcap \ldots \sqcap D_{(j-1)} \sqcap D_{(j+1)} \sqcap \ldots \sqcap D_n) \sqcap (A_1 \sqcap \ldots \sqcap A_{(j-1)} \sqcap A_{(j+1)} \sqcap \ldots \sqcap A_n) \sqcap D_i \sqcap A_i$.
Q' and Q'' are not equivalent because A_i does not subsume $A_1 \sqcap \ldots \sqcap A_{(j-1)} \sqcap A_{(j+1)} \sqcap \ldots \sqcap A_n$.
Indeed A_i is an atomic concept associated with a single view V_i that occurs only once in Q'.
Then there exists Q'' s.t. $Q'' \not\equiv Q'$ and $Q' \sqsubseteq Q'' \sqsubseteq Q$. Therefore Q' cannot be maximally-contained in Q.

($\Leftarrow$) We have to show that
if for all $Q'' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_{n-1}}$, s.t. $\{V_{i_1}, ..., V_{i_{n-1}}\} \subseteq \{V_1, ..., V_n\}$, we have $Q'' \not\sqsubseteq Q$,
then $Q' \equiv V_1 \sqcap \ldots V_n$ is a maximally-contained rewriting of Q.
In other words, we have to show that there is no Q'' s.t. $Q' \sqsubseteq Q''$ and $Q'' \sqsubseteq Q$ with $Q' \not\equiv Q''$.
We refer by (*) the following assumption: for all $Q'' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_{n-1}}$, s.t. $\{V_{i_1}, ..., V_{i_{n-1}}\} \subseteq \{V_1, ..., V_n\}$, we have $Q'' \not\sqsubseteq Q$.
According to hypothesis of lemma 2 $Q' \equiv V_1 \sqcap \ldots V_n$ is subsumed by Q.
Assume that Q' is not a maximally-contained rewriting of Q. Then there exists a conjunction of views Q_1 s.t. $Q' \sqsubseteq Q_1 \sqsubseteq Q$ et $Q' \not\equiv Q_1$.
According to lemma 4, Q_1 subsumes strictly Q' if Q_1 refers only a strict subset of views occurring in Q' and in this case, $Q_1 \not\equiv Q'$.
Let $Q' \equiv V_1 \sqcap ... \sqcap V_n$ et $Q_1 \equiv V_1 \sqcap ... \sqcap V_{j-1} \sqcap V_{j+1} \sqcap ... \sqcap V_n$.
$Q_1 \equiv (D_1 \sqcap \ldots \sqcap D_{(j-1)} \sqcap D_{(j+1)} \sqcap \ldots \sqcap D_n) \sqcap (A_1 \sqcap \ldots \sqcap A_{(j-1)} \sqcap A_{(j+1)} \sqcap \ldots \sqcap A_n)$
and $Q' \equiv (D_1 \sqcap \ldots \sqcap D_{(j-1)} \sqcap D_{(j+1)} \sqcap \ldots \sqcap D_n) \sqcap (A_1 \sqcap \ldots \sqcap A_{(j-1)} \sqcap A_{(j+1)} \sqcap \ldots \sqcap A_n) \sqcap D_i \sqcap A_i$.
and A_i is an atomic concept.
Therefore there exists Q_1 formed with a subset of $\{V_1, ..., V_n\}$ and that is subsumed by Q. Then each conjunction of views that uses supersets of views from Q_1 is subsumed by Q. That contradicts the assumption (*).
Therefore Q' is a maximally-contained rewriting of Q.

C A Bucket-based algorithm for $\mathcal{ALN}(\mathcal{O}_v)$ mediation system

C.1 Bucket algorithm for $\mathcal{ALN}(\mathcal{O}_v)$

Now we give proof of lemma 3:

Proof (proof of lemma 3).
We have $Q' \equiv V_{i_1} \sqcap \ldots \sqcap V_{i_k}$ s.t. Q' is maximally-contained in Q.
Therefore, according to lemma 2, Q' is formed of a minimal subset of views s.t. $Q' \sqsubseteq Q$.

- If $P \in \{A, \neg A\}$ then according to theorem 1, one of the views contains $\forall w.P$. Since the set $\{V_{i_1}, \ldots V_{i_k}\}$ is minimal, one view is sufficient to rewrite $Q \equiv \forall w.P$ and $k = 1$.

– If $P = (\geq nr)$ then according to theorem 1, one of the views contains $\forall w.(\geq mr)$, with $m \geq n$. Since the set $\{V_{i_1}, \ldots V_{i_k}\}$ is minimal, one view is sufficient to rewrite $Q \equiv \forall w.P$ and $k = 1$.
– If $P = (\leq\ n\, r)$, then according to theorem 1, one of the views contains $\forall w.(\leq\ m\, r)$ with $m \leq n$. As above, one view is sufficient to rewrite $Q \equiv \forall w.P$ and $k = 1$.
– If $P = (\leq\ n\, r)$ and $r \in \mathcal{R}_v$ (*), according to normalization rules 8) and 9) and theorem 1, we can find $(k \geq 1)$ views that contain respectively a conjunct $\forall w.r.E_{i_j}$ s.t. $\cap_{j=1}^{k} E_{i_j} \subseteq E'$ and $card(E') \leq n$.
The worst case occurs when the E_{i_j}'s have in pairs a single distinct value and, we have to infer $\forall w.(\leq\ n\, r)$, with $n = 0$. If the maximal number of values in the E_{i_j}'s is l then in worst case, $(l + 1)$ sets of values are necessary to obtain the empty set. Therefore if $r \in \mathcal{R}_v$ and $P = (\leq\ n\, r)$, in the worst case $(l + 1)$ views are necessary to rewrite $Q \equiv \forall w.(\leq\ n\, r)$ and $1 \leq k \leq (l + 1)$.
– If $P = E$, according to normalization rule 8 and theorem 1, $k \geq 1$ views contain respectively a conjunct $\forall w.E_{i_j}$ such that $\cap_{j=1}^{k} E_{i_j} \subseteq E'$ and $E' \subseteq E$. The worst case occurs when E_{i_j} have in pairs a single distinct value, and we have to infer $\forall w.E$, with $E = \emptyset$. If the maximal number of values in the E_{i_j}'s is l then $(l + 1)$ sets of values are necessary to obtain the empty set. Therefore if $P = E$, in the worst case $(l + 1)$ views are necessary to rewrite $Q \equiv \forall w.E$ and $1 \leq k \leq (l + 1)$.
– otherwise, according to theorem 1, $Q' \sqsubseteq \leq\ 0\, r_1$ with r_1 a prefix of w. In the worst case, the concept $\leq\ 0\, r_1$ can be derived by a conjunction of $(p + 1)$ views:
$V_{i_1} \sqsubseteq \forall r_1.r_2.\ldots.r_p.(\leq qr)$
$V_{i_2} \sqsubseteq \forall r_1.r_2.\ldots.r_p.(\geq mr)$, with $q < m$
$V_{i_3} \sqsubseteq \forall r_1.r_2.\ldots.r_{p-1}.(\geq 1r_k)$,
...
$V_{i_{(p+1)}} \sqsubseteq \forall r_1.(\geq 1r_2)$.
This sequence of concepts has been pointed out in [18]. However we can also obtain a conjunction of views that as V_{i_1}, is subsumed by $\forall w'.r_1.r_2.\ldots.r_p.(\leq\ q\, r)$, if $r \in \mathcal{R}_v$, as seen in (*). Such conjunction of views can take the place of view V_{i_1} if such view V_{i_1} does not exist. Therefore, at most $l + 1 + p + 1$ views are necessary to obtain a rewriting $Q' \sqsubseteq \leq\ 0\, r_1$. Hence if $w \in \mathtt{E}(Q')$, in the worst case, $(1 \leq k \leq l + p + 2)$ views are necessary to obtain a rewriting $Q' \sqsubseteq \forall w'.(\leq\ 0\, v)$

C.2 Max-rewritings generation

Proof (Proof of theorem 3).

Let Q' be a maximally-contained rewriting of Q built with the elements of the Q's buckets and $T_{Q'}$ the set of views forming Q'. Assume that T'_Q is not a minimal transversal of $\mathcal{H}_B$. Therefore there exists a minimal transversal $T_{Q''}$ in $\mathcal{H}_B$ such that $T_{Q''} \subset T_{Q'}$. As $T_{Q''}$ meets every edge in E_B and thus every bucket of Q, the conjunction Q'' of views from $T_{Q''}$ is a rewriting of Q and Q' cannot be a maximally-contained rewriting of Q.

D Query rewriting algorithm in $\mathcal{ALN}(\mathcal{O}_v)$ using *iZi*

We give now the proof of property 1.

Proof. Let $X \in \mathcal{L}_{S_1(E,w)}$ s.t. $P_{S_1(E,w)}(E, X)$ is true.
We have to prove that for all $Y \in \mathcal{L}_{S_1(E,w)}$ s.t. $Y \subseteq X$, $P_{S_1(E,w)}(E, Y)$ is true.
Let $X = \{E_{i_1}, ..., E_{i_n}\}$, $Y = \{E_{j_1}, ..., E_{j_k}\}$, with $Y \subseteq X$
As $P_{S_1(E,w)}(E, X)$ is true, $\cap_{j=1}^{n} E_{i_j} \not\subseteq E$.
As $Y \subseteq X$ then $\cap_{j=1}^{n} E_{i_j} \subseteq \cap_{q=1}^{k} E_{j_q}$,
and thus $\cap_{j=1}^{n} E_{i_j} \not\subseteq E$ or equivalently, $P_{S_1(E,w)}(E, Y)$ is true.

We give now the proof of Theorem 4.

Proof. Let $IEE = \{X \in \mathcal{L}_{S_1(E,w)}$ s.t. $P_{S_1(E,w)}(E, X)$ is true $\}$. $\mathcal{B}d^+(IEE)$ gathers the most specialized true patterns. The negative border $\mathcal{B}d^-(IEE)$ gathers the most generalized false patterns, i.e., the minimal subsets of views whose intersection of their restricted set of values for the word w is included in E, that is equivalent to $S_1(E, w)$.

Let us given the proof of property 2.

Proof. Let $X \in \mathcal{L}_{S_2(n,w.R_v)}$ s.t. $P_{S_2(n,w.R_v)}(E, X)$ is true.
We have to prove that for all $Y \in \mathcal{L}_{S_2(n,w.R_v)}$ s.t. $Y \subseteq X$, $P_{S_2(n,w.R_v)}(E, Y)$ is true.
Let $X = \{E_{i_1}, ..., E_{i_n}\}$, $Y = \{E_{j_1}, ..., E_{j_k}\}$, with $Y \subseteq X$.
As $P_{S_2(n,w.R_v)}(E, X)$ is true, $|\cap_{j=1}^{n} E_{i_j}| > n$.
As $Y \subseteq X$ then $\cap_{j=1}^{n} E_{i_j} \subseteq \cap_{q=1}^{k} E_{j_q}$,
and $|\cap_{q=1}^{k} E_{i_j}| > n$ and thus $P_{S_2(n,w.R_v)}(E, Y)$ is true.

Now follows the proof of theorem 5.

Proof. Let $IEN = \{X \in \mathcal{L}_{S_2(n,w.R_v)}$ s.t. $P_{S_2(n,w.R_v)}(E, X)$ is true $\}$. $\mathcal{B}d^+(IEN)$ gathers the most specialized true patterns. The negative border $\mathcal{B}d^-(IEN)$ gathers the most generalized false patterns, i.e., the minimal subsets of views whose cardinality of the intersection of their restricted set of values for the word $w.R_v$ is smaller or equal to n, that is equivalent to $S_E(n, w.R_v)$.

Let us given the proof of property 3.

Proof. Let $X \in \mathcal{L}_{\mathcal{T}_{\mathcal{H}_B}}$ s.t. $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, X)$ is true.
We have to prove that for all $Y \in \mathcal{L}_{\mathcal{T}_{\mathcal{H}_B}}$ s.t. $Y \subseteq X$, $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, Y)$ is true.
Let $Y \in \mathcal{L}_{\mathcal{T}_{\mathcal{H}_B}}$, with $Y \subseteq X$.
As $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, X)$ is true, $\exists B_i \in E_B$ s.t. $X \cap B_i = \emptyset$.
As $Y \subseteq X$ then $B_i \cap Y = \emptyset$,
and thus $P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, Y)$ is true.

Now follows the proof of theorem 6.

Proof. Let $NT = \{X \in \mathcal{L}_{\mathcal{T}_{\mathcal{H}_B}}$ s.t.$P_{\mathcal{T}_{\mathcal{H}_B}}(\mathcal{H}_B, X)$ is true $\}$. By definition, the negative border $\mathcal{B}d^-(NT)$ gathers the most generalized false patterns w.r.t. set inclusion, i.e., the minimal subsets of views which are transversal in $\mathcal{H}_B$, that is equivalent to $\mathcal{T}_{\mathcal{H}_B}$.