DL-LITE$_\mathcal{R}$ in the Light of Propositional Logic for Decentralized Data Management

N. Abdallah and **F. Goasdoué**
LRI: Univ. Paris-Sud, CNRS, and INRIA
{nada,fg}@lri.fr

M.-C. Rousset
LIG: Univ. of Grenoble, CNRS, and INRIA
Marie-Christine.Rousset@imag.fr

Abstract

This paper provides a decentralized data model and associated algorithms for peer data management systems (PDMS) based on the DL-LITE$_\mathcal{R}$ description logic. Our approach relies on reducing query reformulation and consistency checking for DL-LITE$_\mathcal{R}$ into reasoning in propositional logic. This enables a straightforward deployment of DL-LITE$_\mathcal{R}$ PDMSs on top of SomeWhere, a scalable propositional peer-to-peer inference system. We also show how to use the state-of-the-art Minicon algorithm for rewriting queries using views in DL-LITE$_\mathcal{R}$ in the centralized and decentralized cases.

1 Introduction

Ontologies are the backbone of the Semantic Web by providing a conceptual view of data and services made available worldwide through the Web. Description logics are the formal foundations of the OWL ontology web language recommended by W3C. They cover a broad spectrum of logical languages for which reasoning is decidable with a computational complexity varying depending on the set of constructors allowed in the language. Answering conjunctive queries over ontologies is a reasoning problem of major interest for the Semantic Web the associated decision problem of which is not reducible to (un)satisfiability checking. The DL-Lite family [Calvanese *et al.*, 2007] has been specially designed for guaranteeing query answering to be polynomial in data complexity. This is achieved by a query reformulation approach which (1) computes the most general conjunctive queries which, together with the axioms in the Tbox, entail the initial query and (2) evaluates each of those query reformulations against the Abox seen as a relational database. Such an approach has the practical interest that it makes possible to use an SQL engine for the second step, thus taking advantage of well-established query optimization strategies supported by standard relational data management systems. The reformulation step is necessary for guaranteeing the completeness of the answers but is a reasoning step independent of the data. A major result in [Calvanese *et al.*, 2007] is that DL-LITE$_\mathcal{R}$ is one of the maximal fragments of the DL-Lite family supporting tractable query answering over large amounts of data. DL-LITE$_\mathcal{R}$ is a fragment of OWL-DL[1] which extends RDFS[2] with interesting contructors such as inverse roles and disjointness between concepts and between roles. RDFS is the first standard of the W3C concerning the Semantic Web. Its use for associating semantic metadata to web resources is rapidly spreading at a large scale, as shown by the Billion Triple Track of the Semantic Web Challenge (http://challenge.semanticweb.org/).

For scalability and robustness but also for data protection, it is important to investigate a fully decentralized model of the Semantic Web, viewed as a huge peer data management system (PDMS). Each peer may have its own local ontology for describing its data, and interacts with some other peers by establishing mappings with their ontologies. The result is a network of peers with no centralized knowledge and thus no global control on the data and knowledge distributed over the web.

The contribution of this paper is a decentralized data model and associated algorithms for data management in the Semantic Web based on distributed DL-LITE$_\mathcal{R}$. We revisit the centralized current approach of [Calvanese *et al.*, 2007] for data consistency checking and query answering by reformulation in order to design corresponding decentralized algorithms. We also extend the current work on DL-Lite by providing both a centralized and a decentralized algorithm for rewriting queries using views when queries and views are conjunctive queries over DL-LITE$_\mathcal{R}$ ontologies.

Our approach relies on reducing the above data management problems for DL-LITE$_\mathcal{R}$ into decentralized reasoning in distributed propositional logic, in order to deploy DL-LITE$_\mathcal{R}$ PDMSs on top of the SomeWhere platform. SomeWhere is a propositional P2P inference system for which experiments have demonstrated the scalability [Adjiman *et al.*, 2006].

The paper is organized as follows. In Section 2, we present the distributed DL-LITE$_\mathcal{R}$ data model which is based on bridging distributed DL-LITE$_\mathcal{R}$ ontologies with mappings. In Section 3, we provide decentralized algorithms for query answering by reformulation and for data consistency checking. In Section 4, we investigate the problem of query rewriting using views in DL-LITE$_\mathcal{R}$ in the centralized and decentralized cases. Finally, we conclude with a short discussion on related work in Section 5.

[1] http://www.w3.org/2004/OWL/
[2] http://www.w3.org/TR/rdf-schema/

2 Distributed DL-LITE$_\mathcal{R}$

DL-LITE$_\mathcal{R}$ concepts and roles are of the following form:
$B \rightarrow A \mid \exists R,\ C \rightarrow B \mid \neg B,\ R \rightarrow P \mid P^-,\ E \rightarrow R \mid \neg R$
where A denotes an *atomic concept*, P an *atomic role*, and P^- the *inverse* of P. B denotes a *basic concept* (i.e., an atomic concept A or an *unqualified existential quantification on a basic role* $\exists R$) and R a *basic role* (i.e., an atomic role P or its inverse P^-). Finally, C denotes a *general concept* (i.e., a basic concept or its negation) and E a *general role* (i.e., a basic role or its negation).

An *interpretation* $I = (\Delta^I, .^I)$ consists of a nonempty *interpretation domain* Δ^I and an *interpretation function* $.^I$ that assigns a subset of Δ^I to each atomic concept, and a binary relation over Δ^I to each atomic role. The semantics of non atomic concepts and roles is defined as follows:
$(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$
$(\exists R)^I = \{o_1 \mid \exists o_2\ (o_1, o_2) \in R^I\}$
$(\neg B)^I = \Delta^I \backslash B^I$ and $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$
An interpretation I is a *model of a concept* C *(resp. a role* E*)* if $C^I \neq \emptyset$ (resp. $E^I \neq \emptyset$).

DL-LITE$_\mathcal{R}$ **knowledge bases.** A DL-LITE$_\mathcal{R}$ knowledge base (KB) is made of a *Tbox* representing a conceptual view of the domain of interest (i.e., an ontology), and either an *Abox* (a local set of facts) [Calvanese *et al.*, 2007] or *view extensions* (predefined queries over the Tbox together with their answers) [Calvanese *et al.*, 2008b] for representing the data.

A DL-LITE$_\mathcal{R}$ Tbox is a finite set of inclusion statements of the form $B \sqsubseteq C$ and/or $R \sqsubseteq E$. General concepts or roles are only allowed on the right hand side of inclusion statements whereas only basic concepts or roles may occur on the left hand side of inclusion statements. Inclusions of the form $B_1 \sqsubseteq B_2$ or of the form $R_1 \sqsubseteq R_2$ are called *positive inclusions (PIs)*, whereas inclusions of the form $B_1 \sqsubseteq \neg B_2$ or of the form $R_1 \sqsubseteq \neg R_2$ are called *negative inclusions (NIs)*. An interpretation $I = (\Delta^I, .^I)$ is a *model of an inclusion* $B \sqsubseteq C$ (resp. $R \sqsubseteq E$) if $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). It is a *model of a Tbox* if it satisfies all of its inclusion statements. A Tbox $\mathcal{T}$ *logically entails* an inclusion statement α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ is a model of α.

A DL-LITE$_\mathcal{R}$ Abox consists of a finite set of membership assertions on atomic concepts and roles of the form $A(a)$ and $P(a, b)$, stating respectively that a is an instance of A and that the pair of constants (a, b) is an instance of P. The interpretation function of an interpretation $I = (\Delta^I, .^I)$ is extended to constants by assigning to each constant a a distinct object $a^I \in \Delta^I$ (i.e., the so called *unique name assumption* holds). An interpretation I is a *model of the membership assertion* $A(a)$ (resp. $P(a, b)$) if $a^I \in A^I$ (resp., $(a^I, b^I) \in P^I$). It is a *model of an Abox* if it satisfies all of its assertions.

When the extensional knowledge is modeled using view extensions, the KB is of the form $\langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ such that $\mathcal{E}$ is a set of facts of the form $v(\bar{t})$ where v is a *view* of $\mathcal{V}$.

Queries and views over a DL-LITE$_\mathcal{R}$ **KB.** We consider (unions of) *conjunctive queries* of the form $q(\bar{x}) : \exists \bar{y}\ conj(\bar{x}, \bar{y})$ where $conj(\bar{x}, \bar{y})$ is a conjunction of atoms, the variables of which are *only* the free variables $\bar{x}$ and the existential variables $\bar{y}$, and the predicates of which are either *atomic* concepts or roles of the KB. The *arity* of a query is the number of its free variables, e.g., 0 for a *boolean query*.

Given an interpretation $I = (\Delta^I, .^I)$, the semantics q^I of a boolean query q is defined as $true$ if $[\exists \bar{y}\ conj(\emptyset, \bar{y})]^I = true$, and $false$ otherwise, while the semantics q^I of a query q of arity $n \geq 1$ is the relation of arity n defined on (Δ^I) as follows: $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\exists \bar{y}\ conj(\bar{e}, \bar{y})]^I = true\}$.

A *view* v is defined by a query $v(\bar{x}) : \exists \bar{y}\ conj(\bar{x}, \bar{y})$, and has an extension $\mathcal{E}(v)$ which is a set of facts of the form $v(\bar{t})$.

Following the *open world assumption*, we adopt the *sound* semantics, i.e., for every interpretation I for each $v(\bar{t}) \in \mathcal{E}(v)$, $\bar{t}^I \in v^I$.

A *model of a KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (resp. $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$) is an interpretation I that is a model of both $\mathcal{T}$ and $\mathcal{A}$ (resp. of $\mathcal{T}$, $\mathcal{V}$ and $\mathcal{E}$). A KB $\mathcal{K}$ is *consistent* if it has at least one model.

$\mathcal{K}$ *logically entails* a membership assertion β, written $\mathcal{K} \models \beta$, if every model of $\mathcal{K}$ is a model of β.

(Certain) answers of a query over a DL-LITE$_\mathcal{R}$ **KB.** For defining the answers of a query over a KB, it is needed to distinguish the case where the extensions of the query predicates are given in an Abox, from the case where they just can be (partially) inferred from extensions of views. In the latter case, they are called the *certain answers*.

The answer set of a non boolean query q over $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is defined as: $ans(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$ where $\mathcal{C}$ is the set of the constants appearing in the KB, and $q(\bar{t})$ is the closed formula obtained by replacing in the query definition the free variables in $\bar{x}$ by the constants in $\bar{t}$.

The certain answer set of a non boolean query q over $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$, is defined as: $cert(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$.

By convention, the (certain) answer set of a boolean query is $\{()\}$, $()$ is the empty tuple, if $\mathcal{K} \models q()$, and $\emptyset$ otherwise.

DL-LITE$_\mathcal{R}$ **PDMSs** A DL-LITE$_\mathcal{R}$ PDMS $\mathcal{S}$ is a set of peers $\{\mathcal{P}_i\}_{i=1..n}$, where the index i models the identifier of the peer $\mathcal{P}_i$ (e.g., its IP address). Each peer $\mathcal{P}_i$ manages its own DL-LITE$_\mathcal{R}$ KB $\mathcal{K}_i$ written in terms of its own *vocabulary*, i.e., atomic concepts and roles. We will note A_i (resp. P_i) the atomic concept A (resp. the atomic role P) of $\mathcal{P}_i$.

Mappings are here inclusion assertions (PIs and/or NIs) involving concepts and/or roles of two different peers. For simplifying the presentation, we consider that mappings are in both KBs.

From a logical viewpoint, a PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ is a standard (yet distributed) DL-LITE$_\mathcal{R}$ KB $\mathcal{K} = \bigcup_{i=1}^n \mathcal{K}_i$, i.e., in contrast with other approaches ([Calvanese *et al.*, 2008a], [Franconi *et al.*, 2004], [Serafini *et al.*, 2005]) we adopt a standard logical semantics for the mappings.

3 Decentralized Query Answering

We first recall the $Answer$, $Consistent$, and $PerfectRef$ algorithms of [Calvanese *et al.*, 2007] that are used for answering queries over a DL-LITE$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in the centralized case (Section 3.1). Then we provide their decentralized versions (in Sections 3.3 and 3.4). They are based on the propositional encoding summarized in Section 3.2 and the use of the DeCa algorithm [Adjiman *et al.*, 2006] which is the decentralized algorithm for propositional reasoning implemented in the SomeWhere platform.

3.1 Existing DL-LITE$_\mathcal{R}$ algorithms: reminder

Given a union of conjunctive queries Q over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $Answer$ (Algorithm 1) first checks whether $\mathcal{K}$ is inconsistent (line 1). In that case, it returns all the tuples of the arity of Q that can be generated from the constants occurring in $\mathcal{A}$ (line 2). Otherwise, it gets $ans(Q, \mathcal{K})$ by evaluating against $\mathcal{A}$ considered as a relational database the union of conjunctive queries obtained by reformulation of Q (line 3).

Algorithm 1: The original $Answer$ algorithm
$Answer(Q, \mathcal{K})$
Input: a union of conjunctive queries Q and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: $ans(Q, \mathcal{K})$
(1) **if** not $Consistent(\mathcal{K})$
(2) **return** $Alltup(Q, \mathcal{K})$
(3) **else return** $(\bigcup_{q_i \in Q} PerfectRef(q_i, \mathcal{T}))^{db(\mathcal{A})}$

$Consistent$ (Algorithm 2) builds a boolean query q_{unsat} that checks that the DL-LITE$_\mathcal{R}$ formulae that must be disjoint, according to the intentional knowledge modeled in $\mathcal{T}$, indeed have disjoint instances in $\mathcal{A}$. q_{unsat} is obtained as the union of the first-order logic (FOL) translations of the NI-closure of $\mathcal{T}$, denoted $cln(\mathcal{T})$, i.e., the set of all the NIs entailed by $\mathcal{T}$. The FOL translations of NIs are defined by:
$\delta(B_1 \sqsubseteq \neg B_2) = \exists x\ \gamma_1(x) \wedge \gamma_2(x)$ such that
$\gamma_i(x) = A_i(x)$ if $B_i = A_i$
$\gamma_i(x) = \exists y_i\ P_i(x, y_i)$ if $B_i = \exists P_i$
$\gamma_i(x) = \exists y_i\ P_i(y_i, x)$ if $B_i = \exists P_i^-$
$\delta(R_1 \sqsubseteq \neg R_2) = \exists x, y\ \rho_1(x, y) \wedge \rho_2(x, y)$ such that
$\rho_i(x, y) = P_i(x, y)$ if $R_i = P_i$
$\rho_i(x, y) = P_i(y, x)$ if $R_i = P_i^-$

Algorithm 2: The original $Consistent$ algorithm
$Consistent(\mathcal{K})$
Input: a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
Output: $true$ if $\mathcal{K}$ is satisfiable, $false$ otherwise
(1) $q_{unsat} = \bot$ (i.e., q_{unsat} is $false$)
(2) **foreach** $\alpha \in cln(\mathcal{T})$
(3) $q_{unsat} = q_{unsat} \vee \delta(\alpha)$
(4) **if** $q_{unsat}^{db(\mathcal{A})} = \emptyset$
(5) **return** $true$
(6) **else return** $false$

Finally, $PerfectRef$ (Algorithm 3) reformulates each conjunctive query q in Q by using the PIs in $\mathcal{T}$ as rewriting rules. PIs are seen as logical rules that can be applied in backward-chaining to query atoms in order to expand them. More specifically, a PI I is *applicable to an atom* $A(x)$ of a query if I has A in its right-hand side, and a PI I is *applicable to an atom* $P(x_1, x_2)$ of a query if (i) $x_2 = \_$ and the right-hand side of I is $\exists P$; or (ii) $x_1 = \_$ and the right-hand side of I is $\exists P^-$; or (iii) I is a role inclusion assertion and its right-hand side is either P or P^-. Note that $\_$ denotes here an unbounded existential variable of a query.

The following definition (Definition 32 from [Calvanese *et al.*, 2007]) defines the result $gr(g, I)$ of the (backward) application of the PI I to the atom g, which is the core of $PerfectRef$ (loop (a), lines 5 to 7).

Definition 1 (Backward application of a PI to an atom) *Let I be an inclusion assertion that is applicable to the atom g. Then, $gr(g, I)$ is the atom defined as follows:*
- if $g = A(x)$ and $I = A_1 \sqsubseteq A$, then $gr(g, I) = A_1(x)$
- if $g = A(x)$ and $I = \exists P \sqsubseteq A$, then $gr(g, I) = P(x, \_)$
- if $g = A(x)$ and $I = \exists P^- \sqsubseteq A$, then $gr(g, I) = P(\_, x)$
- if $g = P(x, \_)$ and $I = A \sqsubseteq \exists P$, then $gr(g, I) = A(x)$
- if $g = P(x, \_)$ and $I = \exists P_1 \sqsubseteq \exists P$, then $gr(g, I) = P_1(x, \_)$
- if $g = P(x, \_)$ and $I = \exists P_1^- \sqsubseteq \exists P$, then $gr(g, I) = P_1(\_, x)$
- if $g = P(\_, x)$ and $I = A \sqsubseteq \exists P^-$, then $gr(g, I) = A(x)$
- if $g = P(\_, x)$ and $I = \exists P_1 \sqsubseteq \exists P^-$, then $gr(g, I) = P_1(x, \_)$
- if $g = P(\_, x)$ and $I = \exists P_1^- \sqsubseteq \exists P^-$, then $gr(g, I) = P_1(\_, x)$
- if $g = P(x_1, x_2)$ and either $I = P_1 \sqsubseteq P$ or $I = P_1^- \sqsubseteq P^-$ then $gr(g, I) = P_1(x_1, x_2)$
- if $g = P(x_1, x_2)$ and either $I = P_1 \sqsubseteq P^-$ or $I = P_1^- \sqsubseteq P$ then $gr(g, I) = P_1(x_2, x_1)$

The subtle point of $PerfectRef$ is the need of simplifying the produced reformulations (loop (b), lines 8 to 10), so that some PIs that were not applicable to a reformulation become applicable to its simplifications. A simplification amounts to unify two atoms of a reformulation using their *most general unifier* (using $reduce$, line 10) and then to switch the possibly new unbounded existential variables to $\_$ (using τ, line 10).

Algorithm 3: The original $PerfectRef$ algorithm
$PerfectRef(q, \mathcal{T})$
Input: a conjunctive query q and a Tbox $\mathcal{T}$
Output: a union of conjunctive queries
(1) $PR := \{q\}$
(2) **repeat**
(3) $PR' := PR$
(4) **foreach** $q \in PR'$
(5) (a) **foreach** $g \in q$
(6) **if** I is applicable to g
(7) $PR := PR \cup \{q[g/gr(g, I)]\}$
(8) (b) **foreach** $g_1, g_2 \in q$
(9) **if** g_1 and g_2 unify
(10) $PR := PR \cup \{\tau(reduce(q, g_1, g_2))\}$
(11) **until** $PR' = PR$

3.2 Propositional encoding of a DL-LITE$_\mathcal{R}$ Tbox

The propositional encoding of a DL-LITE$_\mathcal{R}$ Tbox $\mathcal{T}$, denoted $\Phi(\mathcal{T})$, is the formula of propositional logic (PL) that corresponds to the union of the PL encoding of every inclusion assersion I of $\mathcal{T}$: $\Phi(\mathcal{T}) = \bigcup_{I \in \mathcal{T}} \Phi(I)$.

The PL encoding of a concept inclusion $B \sqsubseteq C$, denoted $\Phi(B \sqsubseteq C)$ is inductively defined by $\{\Phi(B) \Rightarrow \Phi(C)\}$ where $\Phi(B) = A$ when $B = A$, $\Phi(B) = P^\exists$ when $B = \exists P$, $\Phi(B) = P^{\exists^-}$ when $B = \exists P^-$, $\Phi(C) = \Phi(B)$ when $C = B$, and $\Phi(C) = \neg\Phi(B)$ when $C = \neg B$.

The PL encoding of a role inclusion $R \sqsubseteq E$, denoted $\Phi(R \sqsubseteq E)$, is defined as follows:
$\Phi(P \sqsubseteq Q) = \{P \Rightarrow Q, P^- \Rightarrow Q^-, P^\exists \Rightarrow Q^\exists, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
$\Phi(P^- \sqsubseteq Q) = \{P^- \Rightarrow Q, P \Rightarrow Q^-, P^{\exists^-} \Rightarrow Q^\exists, P^\exists \Rightarrow Q^{\exists^-}\}$
$\Phi(P \sqsubseteq Q^-) = \{P \Rightarrow Q^-, P^- \Rightarrow Q, P^\exists \Rightarrow Q^{\exists^-}, P^{\exists^-} \Rightarrow Q^\exists\}$
$\Phi(P^- \sqsubseteq Q^-) = \{P^- \Rightarrow Q^-, P \Rightarrow Q, P^\exists \Rightarrow Q^\exists, P^{\exists^-} \Rightarrow Q^{\exists^-}\}$
$\Phi(P \sqsubseteq \neg Q) = \{P \Rightarrow \neg Q, P^- \Rightarrow \neg Q^-\}$
$\Phi(P^- \sqsubseteq \neg Q) = \{P^- \Rightarrow \neg Q, P \Rightarrow \neg Q^-\}$
$\Phi(P \sqsubseteq \neg Q^-) = \{P \Rightarrow \neg Q^-, P^- \Rightarrow \neg Q\}$
$\Phi(P^- \sqsubseteq \neg Q^-) = \{P^- \Rightarrow \neg Q^-, P \Rightarrow \neg Q\}$
Note also that in the following $\Phi(E) = \Phi(R)$ when $E = R$, $\Phi(E) = \neg\Phi(R)$ when $E = \neg R$, $\Phi(R) = P$ when $R = P$, and $\Phi(R) = P^-$ when $R = P^-$.

The PL encoding of the distributed Tbox $\bigcup_{i=1}^n \mathcal{T}_i$ of a DL-LITE$_\mathcal{R}$ PDMS is the distributed propositional theory $\bigcup_{i=1}^n \Phi(\mathcal{T}_i)$ obtained by the encoding of each local Tbox $\mathcal{T}_i$.

DECA is a message-based algorithm implemented in the SomeWhere platform ([Adjiman *et al.*, 2006]) which computes in a decentralized manner the logical consequences of propositional clausal theories distributed in a P2P system. More precisely, by a copy of DECA running locally on each peer and transmitting forth and back messages conveying literals and clauses, $DeCA^i(l_i)$ (denoting DECA running on the peer $\mathcal{P}_i$ and triggered with an input literal l_i of the $\mathcal{P}_i$ vocabulary) produces the set of all the *proper prime implicates* of l_i w.r.t. the distributed theory $\bigcup_{i=1}^{n} \Phi(\mathcal{T}_i)$, i.e., the set of prime implicates of $\{l_i\} \cup \bigcup_{i=1}^{n} \Phi(\mathcal{T}_i)$, which are not implicates of $\bigcup_{i=1}^{n} \Phi(\mathcal{T}_i)$ alone.

3.3 Decentralized Consistency Checking

Our approach relies on decentralizing the computation of the NI-closure of a distributed Tbox $\bigcup_{i=1}^{n} \mathcal{T}_i$ of a DL-LITE$_\mathcal{R}$ PDMS without empty roles (the Tbox does not entail a NI $P \sqsubseteq \neg P$) by exploiting a property transfer of the propositional encoding (Theorem 1) and then by using DECA. The subtle point is that in a decentralized setting, we have to launch the computation of the NI-closure from each peer and thus possibly start from local PIs and NIs that could lead to the derivation of new NIs by interacting with NIs and PIs of other peers. For doing so, we define (Definition 2) and compute with DECA the *NI-closure of a peer* w.r.t a PDMS without empty roles.

Theorem 1 (NI-entailment reduced to PL entailment) *Let $\mathcal{T}$ be the distributed Tbox of a* DL-LITE$_\mathcal{R}$ *PDMS without empty roles, and $\Phi(\mathcal{T})$ its PL encoding. Let X and Y be both distinct basic concepts or distinct basic roles:*
$cln(\mathcal{T}) \models X \sqsubseteq \neg Y$ *iff* $\Phi(\mathcal{T}) \models \neg\Phi(X) \vee \neg\Phi(Y)$.

The proof is by induction on the number of rules defining the NI-closure (Definition 9 in [Calvanese *et al.*, 2007]) used for producing $X \sqsubseteq \neg Y$, for the if direction, and on the smallest length of the resolution proof for producing $\Phi(X) \Rightarrow \Phi(\neg Y)$ for the converse direction.

Definition 2 (NI-closure of a peer w.r.t. a PDMS) *Let $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ be the distributed Tbox of a* DL-LITE$_\mathcal{R}$ *PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ without empty roles. The* NI-closure of $\mathcal{P}_i$ w.r.t. $\mathcal{S}$, *denoted $cln(\mathcal{P}_i)$, is obtained from $\Phi(\mathcal{T})$ using* DECA *as follows:*

- *for every PI $Z \sqsubseteq Y \in \mathcal{T}_i$ such that Z is in the vocabulary of $\mathcal{P}_i$ and Y in that of $\mathcal{P}_j$ (j may be i), $Z \sqsubseteq \neg X \in cln(\mathcal{P}_i)$ for any $\neg\Phi(X) \in DeCA^j(\Phi(Y))$.*
- *for every NI $Z \sqsubseteq \neg Y \in \mathcal{T}_i$*
  - *if Z is in the vocabulary of $\mathcal{P}_i$ and Y in that of $\mathcal{P}_j$ (j may be i), $Z \sqsubseteq \neg X \in cln(\mathcal{P}_i)$ for any $\neg\Phi(X) \in DeCA^j(\neg\Phi(Y))$*
  - *if Y is in the vocabulary of $\mathcal{P}_i$ and Z is in that of $\mathcal{P}_j$ (j may be i), $X \sqsubseteq \neg Y \in cln(\mathcal{P}_i)$ for any $\neg\Phi(X) \in DeCA^j(\neg\Phi(Z))$.*

The decentralized version of the original $Consistent$ algorithm, denoted $Consistent^i$ when running on peer $\mathcal{P}_i$, is simply obtained by replacing **foreach** $\alpha \in cln(\mathcal{T})$ in Line 2 of **Algorithm 2** by **foreach** $\alpha \in cln(\mathcal{P}_i)$, and where each conjunctive query of q_{unsat} does not have to be evaluated by $\mathcal{P}_i$ against the (unknown) global Abox of the whole PDMS. Indeed, by construction of q_{unsat}, each of its conjunctive queries has two conjuncts, one from $\mathcal{P}_i$ and another from $\mathcal{P}_j$ (j may be i), the latter providing in its atomic concept or role the identifier j of the peer to contact for the evaluation.

Theorem 2 states the correctness of locally running $Consistent^i$ on each peer $\mathcal{P}_i$ for global consistency checking of a PDMS without empty roles.

Theorem 2 (Correctness of PDMS consistency checking) *Let $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ be a* DL-LITE$_\mathcal{R}$ *PDMS without empty roles. $\mathcal{S}$ is consistent iff $Consistent^i$ returns $true$ for every $i = 1..n$.*

The proof relies first on Theorem 1 showing the equivalence between logical entailement from a Tbox of a NI with entailment in PL of the corresponding propositional encoding. Then, both Lemma 12 in [Calvanese *et al.*, 2007] and the completeness of DECA(proved in [Adjiman *et al.*, 2006]) ensure that $cln(\mathcal{P}_i)$ defined in Definition 2 contains all the NIs entailed by the PDMS and involving a concept or role in the vocabulary of the peer $\mathcal{P}_i$. Finally, it is easy to see that by running $Consistent^i$ for every peer $\mathcal{P}_i$ of the PDMS, we obtain all the NIs entailed by the PDMS. Therefore Theorem 15 and Lemma 16 in [Calvanese *et al.*, 2007] ensure that consistency checking can be made by evaluating the union of conjunctive queries in q_{unsat} against the relevant part of the Abox. It is exactly what running $Consistent^i$ on all the peers in the PDMS does in a decentralized manner.

3.4 Decentralized Query Reformulation

Our approach relies on the propositional encoding and the use of DECA for decentralizing the *backward-closure w.r.t. the PIs* of each atom in the query. Definition 3 defines the *backward-closure* of an atom w.r.t. the PIs as the iteration of the one-step backward application of PIs (Definition 1). Proposition 1 states the termination of this iterative process.

Definition 3 (Backward-closure of an atom w.r.t. PIs) *Let PI be a set of PIs, g an atom, and $\mathcal{A}$ a set of atoms.*
We define the backward-closure *of g w.r.t. PI as $cl\_gr(g, PI) = \bigcup_{i \geq 1} cl\_gr^i(\{g\}, PI)$ where $cl\_gr^i(\{g\}, PI)$ is recursively defined as follows:*

- $cl\_gr^1(\mathcal{A}, PI) = \{gr(g, I) \mid g \in \mathcal{A},\ I \in PI$ *and I is applicable to* $g\}$
- $cl\_gr^{i+1}(\mathcal{A}, PI) = cl\_gr^1(cl\_gr^i(\mathcal{A}, PI), PI)$

Proposition 1 (Termination of backward-closure w.r.t. PIs) *The backward-closure of an atom w.r.t. a set of PIs is finite, i.e., there exists a constant n such that $cl\_gr(g, PI) = \bigcup_{i=1}^{n} cl\_gr^i(\{g\}, PI)$.*

The proof corresponds to the termination proof of $PerfectRef$ (Lemma 34 in [Calvanese *et al.*, 2007]).

Theorem 3 is the equivalent for the PIs of the transfer property of the propositional encoding for the NIs stated in Theorem 1. Its proof is also by induction (number of one-step applications of a PI and smallest length of resolution proofs).

Theorem 3 (Backward-closure reduced to PL entailment) *Let $\mathcal{T}$ be a* DL-LITE$_\mathcal{R}$ *Tbox the PIs of which form the set PI. Let g, g' be atoms, and V_1, V_2 propositional variables. $g' \in cl\_gr(g, PI)$ iff $\Phi(\mathcal{T}) \cup \{\neg V_1\} \models \neg V_2$ where:*

- $g = A(x)$, $g' = A'(x)$, $V_1 = A$, *and* $V_2 = A'$;
- $g = A(x)$, $g' = P(x, \_)$, $V_1 = A$, *and* $V_2 = P^{\exists}$;
- $g = A(x)$, $g' = P(\_, x)$, $V_1 = A$, *and* $V_2 = P^{\exists^-}$;
- $g = P(x, y)$, $g' = Q(x, y)$, $V_1 = P$, *and* $V_2 = Q$;
- $g = P(x, y)$, $g' = Q(y, x)$, $V_1 = P$, *and* $V_2 = Q^-$;
- $g = P(x, \_)$, $g' = A(x)$, $V_1 = P^{\exists}$, *and* $V_2 = A$;
- $g = P(x, \_)$, $g' = Q(x, \_)$, $V_1 = P^{\exists}$, *and* $V_2 = Q^{\exists}$;
- $g = P(x, \_)$, $g' = Q(\_, x)$, $V_1 = P^{\exists}$, *and* $V_2 = Q^{\exists^-}$;
- $g = P(\_, x)$, $g' = A(x)$, $V_1 = P^{\exists^-}$, *and* $V_2 = A$;
- $g = P(\_, x)$, $g' = Q(x, \_)$, $V_1 = P^{\exists^-}$, *and* $V_2 = Q^{\exists}$;
- $g = P(\_, x)$, $g' = Q(\_, x)$, $V_1 = P^{\exists^-}$, *and* $V_2 = Q^{\exists^-}$.

Based on Theorem 3, the decentralized computation of $cl\_gr(g, PI)$ is straighforward using DECA: if g is built from the vocabulary of the peer $\mathcal{P}_i$, $g' \in cl\_gr(g, PI)$ iff $\neg V_2 \in DeCA^i(\neg V_1)$ for the same values of g, g', V_1, and V_2 of the corresponding cases of Theorem 3.

The decentralized version of $PerfectRef$, denoted $PerfectRef^i$ when running on peer $\mathcal{P}_i$, is given in Algorithm 4. For each atom in the query, it computes first (in the decentralized manner explained previously) the set of all of its reformulations, and then it produces a first set of reformulations of the original query by building all the conjunctions between the atomic reformulations (denoted $\mathbb{\wedge}_{i=1}^n cl\_gr(g_i, PI)$ at Line 5). Those reformulations are then possibly simplified by unifying some of their atoms (Lines 8 to 11), and the reformulation process is iterated on those newly produced reformulations until no simplification is possible (general loop starting on Line 4).

Algorithm 4: The decentralized $PerfectRef$ algorithm running on the peer $\mathcal{P}_i$ of the PDMS $\mathcal{S}$

$PerfectRef^i(q)$
Input: a conjunctive query q over the Tbox $\mathcal{T}_i$ of the peer $\mathcal{P}_i$
Output: a union of conjunctive queries over the Tbox $\mathcal{T}$ of the PDMS $\mathcal{S}$

(1) $PR := \{q\}$
(2) $PR' := PR$
(3) **while** $PR' \neq \emptyset$
(4) (a) **foreach** $q' = g_1 \wedge g_2 \wedge \ldots \wedge g_n \in PR'$
(5) $PR'' = \mathbb{\wedge}_{i=1}^n cl\_gr(g_i, PI)$
(6) $PR' = \emptyset$
(7) (b) **foreach** $q'' \in PR''$
(8) **foreach** $g_1', g_2' \in q''$
(9) **if** g_1' and g_2' unify
(10) $PR' := PR' \cup \{\tau(reduce(q'', g_1', g_2'))\}$
(11) $PR = PR \cup PR' \cup PR''$
(12) **return** PR

The following theorem states the correctness of the decentralized reformulation algorithm $PerfectRef^i$.

Theorem 4 (Correctness of $PerfectRef^i$) *Let $\mathcal{T} = \bigcup_{i=1}^n \mathcal{T}_i$ be a Tbox of a PDMS. Let q be a conjunctive query over $\mathcal{T}_i$. $PerfectRef^i(q)$ returns the same set of conjunctive queries as $PerfectRef(q, \mathcal{T})$.*

Its proof results (1) from the observation that the centralized version of $PerfectRef^i$ (in which $cl\_gr(g_i, PI)$ is computed by iterating the one-step application of PIs on each atom g_i of the query) produces the same results than the original $PerfectRef$, and (2) from Theorem 3 and the completeness of DECA, ensuring the decentralized computation of the whole set $cl\_gr(g_i, PI)$.

In contrast with the original $Answer$ algorithm, the global consistency of the PDMS cannot be checked at query time since the queried peer $\mathcal{P}_i$ does not know all the peers in the PDMS. However, it can get the identifiers $id_1, \ldots, id_k$ of the peers involved in a reformulation of the query (to contact them) from the identifiers used in the atomic concept and role names involved in that reformulation. Algorithm 5 describes the decentralized $Answer^i$ algorithm that checks in a decentralized manner whether $\bigcup_{j=1}^k (\mathcal{T}_{id_j} \cup \mathcal{A}_{id_j})$ is consistent and computes the set of corresponding answers by evaluating each reformulation against the relevant Aboxes.

Algorithm 5: The decentralized $Answer$ algorithm running on the peer $\mathcal{P}_i$ of the PDMS $\mathcal{S}$

$Answer^i(Q)$
Input: a union of conjunctive queries Q over the KB $\mathcal{K}_i = \langle \mathcal{T}_i, \mathcal{A}_i \rangle$ of $\mathcal{P}_i$
Output: $ans(Q, \mathcal{K})$ where $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the KB of the PDMS $\mathcal{S}$

(1) $q = \bigcup_{q' \in Q} PerfectRef^i(q')$
(2) **if** $Consistent^{id_j}$ returns true for every peer $\mathcal{P}_{id_j}$ involved in q
(3) **return** $q^{db(\bigcup_{j=1}^k \mathcal{A}_{id_j})}$
(4) **else return** the singleton $\{\bot\}$

In that algorithm $\bot$ replaces $AllTup(Q, \mathcal{K})$ of the original $Answer$ algorithm.

The interest of Algorithm 5 is to provide *well-founded* answers, i.e., answers that can be entailed from a consistent subset of the (possibly inconsistent) KB of the PDMS.

4 Query Answering using Views by Rewriting

We provide algorithms for computing the certain answers of a query over a (centralized or decentralized) DL-LITE$_\mathcal{R}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{E}$ is the extension of views in $\mathcal{V}$ that are conjunctive queries over $\mathcal{T}$. For doing so, we make use of the scalable $MiniCon$ [Pottinger and Halevy, 2001] algorithm which produces the maximally-contained conjunctive rewritings of a conjunctive query q using a set $\mathcal{V}$ of conjunctive views. A *conjunctive rewriting* of q is a conjunctive query q_v whose body predicates are the head predicates of views in $\mathcal{V}$ such that $\mathcal{T} \cup \mathcal{V} \models \forall \bar{x} \, (q_v(\bar{x}) \Rightarrow q(\bar{x}))$. In that setting ([Halevy, 2001]), the set of certain answers of a query can be obtained by evaluating against the view extensions the (finite) union of its maximally-contained conjunctive rewritings.

Centralized Case First $ViewConsistent(\mathcal{K})$ checks the consistency of the KB It is a variant of the original $Consistent$ algorithm obtained by replacing:

- $q_{unsat} = q_{unsat} \vee \delta(\alpha)$ in Line 3 of Algorithm 2 by $q_{unsat} = q_{unsat} \vee MiniCon(\delta(\alpha), \mathcal{V})$, where $MiniCon(\delta(\alpha), \mathcal{V})$ provides the maximally-contained rewritings using $\mathcal{V}$ of the FOL translation $\delta(\alpha)$ of the NI α,
- $q_{unsat}^{db(\mathcal{A})}$ in Line(4) of Algorithm 2 by the evaluation against the view extensions: $q_{unsat}^{db(\mathcal{E})}$.

Algorithm 6 describes the $CertainAnswer$ algorithm. If the KB is inconsistent, the algorithm returns $Alltup(Q, \mathcal{E})$, the set of all the tuples of the arity of Q generated from the constants in $\mathcal{E}$. Otherwise, it computes (using $MiniCon(q', \mathcal{V})$) the rewritings in terms of the views of the conjunctive queries q' returned by $PerfectRef$ as the different ways of unfolding the initial query using the PIs of $\mathcal{T}$.

Algorithm 6: The $CertainAnswer$ algorithm
$CertainAnswer(Q, \mathcal{K})$
Input: a union of conjunctive queries Q and a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$
Output: $cert(Q, \mathcal{K})$
(1) **if** not $ViewConsistent(\mathcal{K})$
(2) **return** $Alltup(Q, \mathcal{E})$
(3) **else**
(4) $Q' = \bigcup_{q \in Q} PerfectRef(q, \mathcal{T})$
(5) **return** $(\bigcup_{q' \in Q'} MiniCon(q', \mathcal{V}))^{db(\mathcal{E})}$

Theorem 5 (Correctness of $CertainAnswer$) *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{V}, \mathcal{E} \rangle$ be a* DL-LITE$_\mathcal{R}$ *consistent KB and Q a union of conjunctive queries. $cert(Q, \mathcal{K}) = CertainAnswer(Q, \mathcal{K})$.*

For one direction, from $q' \in PerfectRef(q, \mathcal{T})$ and $q_v \in Minicon(q', \mathcal{V})$ we infer: $\mathcal{T} \cup \mathcal{V} \models \forall \bar{x} \, (q_v(\bar{x}) \Rightarrow q(\bar{x}))$, and thus q_v is a conjunctive rewriting of q, the evaluation of which provides certain answers. For the converse direction: if $\bar{t}$ is a certain answer, by adapting the notion of witness of a tuple introduced in Lemma 39 of [Calvanese *et al.*, 2007] to $\mathcal{E}$ instead of $\mathcal{A}$, we build from the witness of $\bar{t}$ a specific conjunctive query q_v over view atoms such that $\bar{t}$ is in its answer set and we show by induction that this query is a rewriting of a reformulation q' of q. Since $Minicon$ computes all the maximally-contained rewritings (Theorem 1 in [Pottinger and Halevy, 2001]), q_v is contained in one of them, and $\bar{t}$ will be returned by Line 5 of Algorithm 6.

Decentralized Case For space limitation, we just sketch the approach: the decentralized versions of $ViewConsistent$ and of $CertainAnswer$ are denoted $ViewConsistent^i$ and $CertainAnswer^i$ when running on a peer $\mathcal{P}_i$. They are extensions of the decentralized algorithms $Consistent^i$ and $Answer^i$ presented in the previous section. They use the function $fetchViews(q)$ where q is a conjunctive query possibly involving the vocabulary of different peers: $fetchViews(q)$ retrieves from those peers the views a body atom of which can be unified with a body atom of the query.

The $ViewConsistent^i$ algorithm extends $Consistent^i$ by replacing the evaluation of q_{unsat} against the relevant Aboxes by the evaluation of Q_{unsat} against the relevant view extensions, where Q_{unsat} is obtained from q_{unsat} (which is computed as in $Consistent^i$) as follows:

$Q_{unsat} = \bigcup_{q \in q_{unsat}} MiniCon(q, fetchViews(q))$.

The $CertainAnswer^i$ algorithm extends $Answer^i$ by replacing the evaluation against the relevant Aboxes of the union q of conjunctive queries obtained by reformulation of the initial query (Line 2 in Algorithm 5) by evaluating against the relevant view extensions the following query Q:

$Q = \bigcup_{q' \in q} MiniCon(q', fetchViews(q'))$.

5 Conclusion

This papers builds on and extends existing work in data integration ([Calvanese *et al.*, 2007; 2008b] and [Pottinger and Halevy, 2001]). For view-based query answering in DL-LITE$_\mathcal{R}$ we provide a centralized and a decentralized algorithm to compute the certain answers based on rewritings. For the decentralized case, our work extends the data model of the SOMEOWL and SOMERDFS PDMSs ([Adjiman *et al.*, 2006; 2007]). We follow the same approach based on limiting the data model allowing to reduce consistency checking and query reformulation to reasoning in propositional logic. It is a way of getting the decidability for query answering in PDMS which is not guaranteed in general ([Halevy *et al.*, 2003]), while adopting a standard logical semantics, in contrast with other works (e.g., ([Calvanese *et al.*, 2008a], [Franconi *et al.*, 2004], [Serafini *et al.*, 2005]). It is also a distinguishing point from the approach in [Bertossi and Bravo, 2007] (based on answer set programs) for defining consistent answers in possibly inconsistent PDMSs.

References

[Adjiman *et al.*, 2006] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting. *JAIR*, 25, 2006.

[Adjiman *et al.*, 2007] P. Adjiman, F. Goasdoué, and M.-C. Rousset. Somerdfs in the semantic web. *JODS*, 8, 2007.

[Bertossi and Bravo, 2007] L. E. Bertossi and L. Bravo. The semantics of consistency and trust in peer data exchange systems. In *LPAR*, 2007.

[Calvanese *et al.*, 2007] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *JAR*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2008a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in p2p data integration: An epistemic logic approach. *Information Systems*, 33(4-5), 2008.

[Calvanese *et al.*, 2008b] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. View-based query answering over description logic ontologies. In *KR*, 2008.

[Franconi *et al.*, 2004] E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. Queries and updates in the coDB peer-to-peer database system. In *VLDB*, 2004.

[Halevy *et al.*, 2003] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, 2003.

[Halevy, 2001] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[Pottinger and Halevy, 2001] R. Pottinger and A. Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.

[Serafini *et al.*, 2005] L. Serafini, A. Borgida, and A. Tamilin. Aspects of distributed and modular ontology reasoning. In *IJCAI*, 2005.