Robust Module-based Data Management

François Goasdoué, Marie-Christine Rousset

Robust Module-based Data Management

François Goasdoué, *LRI, Univ. Paris-Sud,* and Marie-Christine Rousset, *LIG, Univ. Grenoble*

Abstract—The current trend for building an ontology-based data management system (DMS) is to capitalize on efforts made to design a preexisting well-established DMS (a *reference* system). The method amounts to extracting from the reference DMS a piece of schema relevant to the new application needs – a module –, possibly personalizing it with extra-constraints w.r.t. the application under construction, and then managing a dataset using the resulting schema.
In this paper, we extend the existing definitions of modules and we introduce novel properties of *robustness* that provide means for checking easily that a robust module-based DMS evolves *safely* w.r.t. both the schema and the data of the reference DMS. We carry out our investigations in the setting of description logics which underlie modern ontology languages, like RDFS, OWL, and OWL2 from W3C. Notably, we focus on the DL-lite$_\mathcal{A}$ dialect of the DL-lite family, which encompasses the foundations of the QL profile of OWL2 (i.e., DL-lite$_\mathcal{R}$): the W3C recommendation for efficiently managing large datasets.

Index Terms—H.1 Models and Principles, H.2 Database Management, H.2.8.k Personalization, I.1.2.b Algorithms for data and knowledge management, I.2 Artificial Intelligence, I.2.12 Intelligent Web Services and Semantic Web.

1 Introduction

In many application domains (e.g., medicine or biology), comprehensive schemas resulting from collaborative initiatives are made available. For instance, SNOMED is an ontological schema containing more than 400.000 concept names covering various areas such as anatomy, diseases, medication, and even geographic locations. Such well-established schemas are often associated with reliable data that have been carefully collected, cleansed, and verified, thus providing *reference* ontology-based data management systems (DMSs) in different application domains.

A good practice is therefore to build on the efforts made to design reference DMSs whenever we have to develop our own DMS with specific needs. A way to do this is to extract from the reference DMS the piece of schema relevant to our application needs, possibly to personalize it with extra-constraints w.r.t. our application under construction, and then to manage our own dataset using the resulting schema.

Recent work in description logics (DLs, [1]) provides different solutions to achieve such a reuse of a reference ontology-based DMS. Indeed, modern ontological languages – like the W3C recommendations RDFS, OWL, and OWL2 – are actually XML-based syntactic variants of well-known DLs. All those solutions consist in extracting a *module* from an existing ontological schema such that all the constraints concerning the *relations of interest* for the application under construction are captured in the module [2]. Existing definitions of modules in the literature basically resort to the notion of (deductive) conservative extension of a schema or of uniform interpolant of a schema, a.k.a. forgetting about *non*-interesting relations of a schema. [3] formalizes those two notions for schemas written in DLs and discusses their connection. Up to now, conservative extension has been considered for defining a module as a *subset* of a schema. In contrast, forgetting has been considered for defining a module as only logically *implied* by a schema (by definition forgetting cannot lead to a subset of a schema in the general case). Both kinds of modules have been investigated in various DLs, e.g., DL-lite [4], [5], $\mathcal{EL}$ [6], [7], [8], and $\mathcal{ALC}$ [7], [9], [10].

In this paper, we revisit the reuse of a reference ontology-based DMS in order to build a new DMS with specific needs. We go one step further by not only considering the *design* of a module-based DMS (i.e., how to extract a module from a ontological schema): we also study how a module-based DMS can benefit from the reference DMS to enhance its own data management skills. We carry out our investigations in the setting of DL-lite, which is the foundation of the QL profile of OWL2 recommended by the W3C for efficiently managing large RDF datasets. RDF is the W3C's Semantic Web data model, which is rapidly spreading in more and more applications, and can be seen as a simple relational model restricted to unary and binary predicates. In addition, DL-lite comes with efficient inference algorithms [11] for querying RDF data through (DL-lite) ontologies and for checking data consistency w.r.t. integrity constraints expressed in DL-lite.

Our contribution is to introduce and study novel properties of *robustness* for modules that provide means for checking easily that a robust module-based DMS evolves *safely* w.r.t. both the schema and the data of the reference DMS. From a module *robust to consistency*

- *F. Goasdoué is with the Laboratoire de Recherche en Informatique (LRI), Bâtiment 490, Université Paris-Sud, 91405 Orsay Cedex, France. E-mail: fg@lri.fr*
- *M.-C. Rousset is with the Laboratoire d'Informatique de Grenoble, Université de Grenoble, 681 rue de la Passerelle, BP 72, 38402 St. Martin d'Heres Cedex, France. E-mail: Marie-Christine.Rousset@imag.fr*

checking, for any data update in a corresponding module-based DMS, we show how to query the reference DMS for checking whether the local update does not bring any inconsistency with the data and the constraints of the reference DMS. From a module *robust to query answering*, for any query asked to a module-based DMS, we show how to query the reference DMS for obtaining additional answers by also exploiting the data stored in the reference DMS.

It is worth noticing that our investigations are sustained by real use-cases. For instance, the MyCF DMS (MyCorporisFabrica, www.mycorporisfabrica.org, [12]) has been built *by hand* from the FMA DMS (Foundational Model of Anatomy, sig.biostr.washington.edu/projects/fm). The extraction step has focused on particular parts of the human body (e.g., hand, foot, and knee), while the personalization step has enriched the descriptions of these parts with both 3D geometrical and bio-mechanical information. Notably, careful attention was paid so that MyCF still conforms with FMA at the end of the *manual* process.

The paper is organized as follows. We start with an illustrative example in Section 2 that highlights the issues and solutions on which we elaborate in the rest of the paper. In Section 3, we present the DL-lite description logic, which provides the formal basis of Section 4, in which we study robust modules and safe personalization. In Section 5, we provide algorithms and complexity results for extracting robust modules from schemas and for checking the safe personalization of modules. We conclude in Section 6 with related work and perspectives.

$\mathcal{O}$:

1) `Publication` $\sqsubseteq \exists$`hasTitle`, (*funct* `hasTitle`)
2) `Publication` $\sqsubseteq \exists$`hasDate`, (*funct* `hasDate`)
3) `Publication` $\sqsubseteq \exists$`hasVenue`, (*funct* `hasVenue`)
4) `Publication` $\sqsubseteq \exists$`hasAuthor`
5) $\exists$`hasTitle` $\sqsubseteq$ `Publication`
6) `ConfPaper` $\sqsubseteq$ `Publication`, `JournPaper` $\sqsubseteq$ `Publication`, `ConfPaper` $\sqsubseteq \neg$`JournPaper`
7) `ShortPaper` $\sqsubseteq$ `ConfPaper`, `FullPaper` $\sqsubseteq$ `ConfPaper`, `FullPaper` $\sqsubseteq \neg$`ShortPaper`, `Survey` $\sqsubseteq$ `JournPaper`

$\mathcal{D}$:

`Publication`

...

`hasTitle`

doi_1	"CAQUMV"
doi_2	"AQUVAS"
doi_3	"MC : ASAAQUV"
...	...

`hasDate`

doi_1	"1998"
doi_2	"2001"
doi_3	"2001"
...	...

`hasVenue`

doi_1	"PODS"
doi_2	"VLDBJ"
doi_3	"VLDBJ"
...	...

`hasAuthor`

doi_1	"SA"
doi_1	"OD"
doi_2	"AH"
doi_3	"AH"
doi_3	"RP"
...	...

`ConfPaper`

...

`JournPaper`

doi_3
...

`ShortPaper`

...

`FullPaper`

doi_1
...

`Survey`

doi_2
...

Fig. 1. A reference DMS defined by the schema $\mathcal{O}$ and the dataset $\mathcal{D}$.

2 Illustrative Example

Consider a reference DMS for scientific publications (like DBLP) defined by the ontological schema $\mathcal{O}$ and the dataset $\mathcal{D}$ in Figure 1.

The schema $\mathcal{O}$ is built upon the unary relations `Publication`, `ConfPaper`, `ShortPaper`, `FullPaper`, `JournPaper`, `Survey`, and the binary relations `hasTitle`, `hasDate`, `hasVenue`, and `hasAuthor`. It consists of inclusion constraints and of integrity constraints (disjointness and functional constraints). These constraints are written in Figure 1 using DL-lite, in which $\exists$`r` denotes the usual (relational) projection on the *first* attribute of the binary relation `r` and (*funct* `r`) denotes the functional dependency from the first attribute of the binary relation `r` to the second one. The constraints in $\mathcal{O}$ state that any publication has a single title [1.], a single date of publication [2.], a single venue [3.], and at least one author [4.]. In addition, only publications have a title [5.], papers in conference proceedings or in journals (which are disjoint) are publications [6.], short papers or full papers (which are disjoint) are papers in conference proceedings, and surveys are journal papers [7.].

The dataset $\mathcal{D}$ consists of instances for the relations in $\mathcal{O}$. It is expressed as relational tables in Figure 1. In particular, those tables state that:

- doi_1 is the *Digital Object Identifier*[1] (DOI) of the full paper entitled "Complexity of Answering Queries Using Materialized Views" and published in PODS'98 by Serge Abiteboul ("SA") and Oliver M. Duschka ("OD"),
- doi_2 is the DOI of the survey entitled "Answering queries using views: A survey" and published in VLDB Journal in 2001 by Alon Y. Halevy ("AH"), and
- doi_3 is the DOI of the journal paper entitled "Mini-Con: A scalable algorithm for answering queries using views" and published in VLDB Journal in 2001 by Rachel Pottinger ("RP") and Alon Y. Halevy ("AH").

It is worth noticing here that in contrast with the relational model, data management in DLs needs some reasoning to exhibit all the relevant implicit data regarding a given task, e.g., consistency checking or query answering. For instance, doi_1 does not have to be explicitly stored in the `ConfPaper` and `Publication` tables due to the inclusion constraints in $\mathcal{O}$, while it implicitly belongs to those relations due to these constraints.

1. http://www.doi.org

$\mathcal{O}'$: JournPaper $\sqsubseteq \exists$hasAuthor

$\mathcal{D}'$:

JournPaper	
	doi_1
	$\cdots$

hasAuthor	
doi_1	"SA"
doi_1	"OD"
$\cdots$	$\cdots$

Fig. 2. A module-based DMS defined by the schema $\mathcal{O}'$ and the dataset $\mathcal{D}'$.

2.1 Designing a module-based DMS

Suppose that we have to develop a DMS about scientific publications, e.g., for a company or a university. If we are interested in managing journal papers and their authors only, we can extract a module from $\mathcal{O}$ w.r.t. the *relations of interest* JournPaper and hasAuthor. A possible module $\mathcal{O}'$ consists of the constraint JournPaper $\sqsubseteq \exists$hasAuthor.

Suppose now that the person in charge of populating this module-based DMS stores by mistake doi_1 in the local JournPaper table, and its authors "SA" and "OD" in the local hasAuthor table, as illustrated in Figure 2.

2.2 Global consistency: illustration

It is easy to see that though our module-based DMS is consistent, it is inconsistent together with the reference DMS: doi_1 is a journal paper in our DMS, while it is a conference paper in the reference DMS. This violates a constraint of the reference schema ([6.] in $\mathcal{O}$).

Detecting this kind of inconsistency, called a *global* inconsistency, is important since it indicates that some of our data contradicts the reference DMS, and thus is probably erroneous. Our basic idea is therefore to use the whole reference DMS (schema and data) as extra-constraints to be satisfied by a module-based DMS. Of course, we do not want to import the whole reference DMS into our own DMS in order to do this. Instead, we extend the notion of module to *robustness to consistency checking*, so that global consistency checking can be performed on demand or upon update: We ensure that the module captures the (possibly implied) constraints in the reference schema that are required to detect inconsistency related to the relations of interest. Then, at global consistency checking time, those constraints are verified against the *distributed* dataset consisting of the dataset of the module-based DMS plus that of the reference DMS.

Making our module $\mathcal{O}'$ robust to consistency checking requires adding (integrity) constraints like JournPaper $\sqsubseteq \neg$FullPaper which allows detecting inconsistency related to the relation of interest JournPaper. Note that this constraint brings the relation FullPaper into the module, while it is not of interest w.r.t. our application needs. At global consistency checking time, the constraints in the module that allow detecting inconsistency w.r.t. the relation of interests are verified by evaluating a boolean union of conjunctive queries $Q()$:- $[\exists x$ JournPaper$(x) \wedge$ FullPaper$(x)] \vee \cdots$ which looks for the existence of counter-examples to any of those constraints. Here, the first conjunctive query in $Q()$ looks for a possible counter-example to JournPaper $\sqsubseteq \neg$FullPaper. The subtle point is that the evaluation of $Q()$ is distributed among the module-based DMS and the reference one. As a result, the query to evaluate against the DMSs is $Q()$:- $[\exists x$ (JournPaper$(x) \vee$ JournPaper$_{\text{ref}}(x)) \wedge$ FullPaper$_{\text{ref}}(x)] \vee \cdots$ where the distributed evaluation is reflected in the names of the relations: r denotes a local relation, while r_{ref} denotes the corresponding relation in the reference DMS. The above $Q()$ exhibits a global inconsistency due to doi_1 belonging to the local JournPaper table of our module-based DMS and to the FullPaper$_{\text{ref}}$ table of the reference DMS.

2.3 Global answers: illustration

Suppose now that our DMS can answer conjunctive queries (a.k.a. select-project-join queries), e.g., $Q(x)$:- JournPaper$(x) \wedge$ hasAuthor$(x,$"AH"$)$ asking for the journal papers written by Alon Y. Halevy. In some situation, it is interesting to provide answers from our DMS together with the reference one, called *global* answers, typically when our own DMS provides no or too few answers. To do so, we extend the notion of module to *robustness to query answering*, so that global query answering can be performed on demand. We ensure that the module captures the knowledge in the reference schema that is required to answer any query built upon the relations of interest. Then, at global query answering time, this knowledge is used to identify the relevant data for a given query within the *distributed* dataset consisting of the dataset of the module-based DMS plus that of the reference DMS.

Making $\mathcal{O}'$ robust to query answering requires adding inclusion constraints like Survey $\sqsubseteq$ JournPaper which allows exhibiting implicit tuples for the relation of interest JournPaper: those explicitly stored for the relation Survey. Again, such a constraint brings the relation Survey into the module, while it is not of interest w.r.t. our application needs. At global query answering time, the contraints in the module that allow answering a given query built upon relations of interest are used to reformulate this query into a union of conjunctive queries $Q(x)$:- $[$JournPaper$(x) \wedge$ hasAuthor$(x,$"AH"$)] \vee$ $[$Survey$(x) \wedge$ hasAuthor$(x,$"AH"$)] \vee \cdots$ which models all the ways to answer it from a dataset. Here, the second conjunctive query in $Q(x)$ results from the inclusion constraint Survey $\sqsubseteq$ JournPaper. Again, since the dataset is distributed among the module-based DMS and the reference one, the query to evaluate in fact is $Q(x)$:-
$[($JournPaper$(x) \vee$ JournPaper$_{\text{ref}}(x))$
$\wedge$ (hasAuthor$(x,$"AH"$) \vee$ hasAuthor$_{\text{ref}}(x,$"AH"$))]$
$\vee$ $[$Survey$_{\text{ref}}(x)$
$\wedge$ (hasAuthor$(x,$"AH"$) \vee$ hasAuthor$_{\text{ref}}(x,$"AH"$))] \vee \cdots$.
In particular, $Q(x)$ finds doi_2 and doi_3 as global answers, due to the presence in the reference DMS of: doi_2 in the Survey$_{\text{ref}}$ table, $(doi_2,$"AH"$)$ in the hasAuthor$_{\text{ref}}$ table, doi_3 in the JournPaper$_{\text{ref}}$ table, and $(doi_3,$"AH"$)$ in the hasAuthor$_{\text{ref}}$ table.

2.4 Safe personalization: illustration

Suppose now that a (possibly robust) module does not meet all the constraints for our application under development. A personalization step – which amounts to adding the appropriate constraints – is thus necessary. However, it must be carefully done since personalizing can lead to loose global data management skills (i.e., robustness) or even the essence of the notion of module. To prevent this, we exhibit sufficient conditions for a *safe personalization*.

For instance, suppose that we personalize $\mathcal{O}'$ with the constraints $\texttt{hasAuthor} \sqsubseteq \texttt{hasRightsOn}^-$ and $\exists\texttt{hasRightsOn}^- \sqsubseteq \texttt{JournPaper}$ in order to express that any author of a journal paper has some rights on that paper, the notion of rights concerning only journal papers. Note that in DLs, $\texttt{r}^-$ denotes the inverse of the binary relation r, i.e., the relation obtained by swapping its two attributes. Thus, $\exists\texttt{r}^-$ denotes the usual (relational) projection on the *second* attribute of r. Adding the above constraints to $\mathcal{O}'$ leads to the implied constraint $\exists\texttt{hasAuthor} \sqsubseteq \texttt{JournPaper}$, which makes sense w.r.t. the reference DMS as it is built upon relations in $\mathcal{O}$ only. Yet, this constraint does not hold in the reference DMS. As a result, the personalization of $\mathcal{O}'$ mentioned above is *not* safe. In fact, the extra-constraint $\exists\texttt{hasAuthor} \sqsubseteq \texttt{JournPaper}$ makes the reference DMS insconsistent: on one hand, conference papers are declared disjoint from journal papers, on the other hand, by having authors they are inferred by the extra-constraint as beeing journal papers, therefore making any dataset including a conference paper inconsistent.

2.5 Reducing data storage: illustration

Robust module-based DMSs offer an interesting peculiarity w.r.t. data storage. Indeed, global data management is performed on a dataset that is distributed among the module-based DMS and the reference one. Notably, redundancy can occur in the distributed dataset when some same instances of the relations of interest are both stored in the module-based DMS and stored – explicitly or implicitly– in the reference DMS. Therefore, a way of reducing data storage in a robust module-based DMS is to store only data that are not already somehow stored in the reference DMS. This can be easily checked by asking queries to this DMS.

For instance, consider a robust version of the module $\mathcal{O}'$ that we safely personalize with the constraints $\texttt{hasContactAuthor} \sqsubseteq \texttt{hasAuthor}$ and $(\mathit{funct}\ \texttt{hasContactAuthor})$ stating that having a *single* contact author is a particular case of having an author. If the purpose of our DMS is *only* to store the contact authors for the journal papers of the reference DMS, the corresponding module-based DMS with minimal storage is depicted in Figure 3. In particular, nothing is stored in the local tables JournPaper and hasAuthor for the sake of non-redundancy: doi_2 and doi_3 are not stored locally in JournPaper because doi_2 is explicitly

$\mathcal{O}''$:
$\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$
$\cdots$
$\texttt{hasContactAuthor} \sqsubseteq \texttt{hasAuthor},$
$(\mathit{funct}\ \texttt{hasContactAuthor})$

$\mathcal{D}''$:

JournPaper	
~~doi_2~~	
~~doi_3~~	
~~$\vdots$~~	

hasAuthor	
~~doi_2~~	~~"AH"~~
~~doi_3~~	~~"AH"~~
~~doi_3~~	~~"RP"~~
~~$\vdots$~~	~~$\vdots$~~

hasContactAuthor	
doi_2	"AH"
doi_3	"RP"
$\cdots$	$\cdots$

Fig. 3. A non-redundant robust module-based DMS defined by the schema $\mathcal{O}''$ and the dataset $\mathcal{D}''$.

stored in $\texttt{JournPaper}_{\text{ref}}$ and doi_3 is implicitly stored in $\texttt{JournPaper}_{\text{ref}}$ since it is explicitly stored in $\texttt{Survey}_{\text{ref}}$ and $\texttt{Survey} \sqsubseteq \texttt{JournPaper}$ holds in $\mathcal{O}$.

It is worth noticing that distributing data storage may be done in order to optimize the distributed evaluation of queries on relations of interest over the module-based DMS and the reference one. For instance, consider the query asking for the authors of journal papers whose contact author is Rachel Pottinger: $Q(x)$:- $\exists y\ \ \texttt{JournPaper}(y) \wedge \texttt{hasAuthor}(y, x) \wedge \texttt{hasContactAuthor}(y, "RP")$. The possible paper identifiers for whom Rachel Pottinger is the contact author are obtained locally from the local table hasContactAuthor: here, we obtain doi_3 only. Verifying that doi_3 is a journal paper and getting its authors is not done locally but by querying the reference DMS. In particular, the fact that doi_3 is a journal paper is obtained by inference from the storage of doi_3 in the reference table $\texttt{Survey}_{\text{ref}}$ and from the constraint $\texttt{Survey} \sqsubseteq \texttt{JournPaper}$ declared in the reference schema. Sharing data storage this way avoids redundancy in the distributed evaluation of the query Q over the module-based DMS and the reference one.

3 DL-LITE DATA MODEL

Generally speaking, in DLs [1], a schema is called a *Tbox* and its associated dataset is called an *Abox*.

A Tbox $\mathcal{T}$ is defined upon a *signature* (a.k.a. *vocabulary*), denoted $\text{sig}(\mathcal{T})$, which is the disjoint union of a set of unary relations called *atomic concepts* and a set of binary relations called *atomic roles*. It consists of a set of constraints called *terminological axioms*, typically inclusion constraints between complex concepts or roles, i.e., unary or binary DL formulae built upon atomic relations using the constructors allowed in DL under consideration.

An Abox defined upon $\text{sig}(\mathcal{T})$ is a set of facts called *assertional axioms*, relating DL formulae to their instances.

A *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is made of a *Tbox* $\mathcal{T}$ and an *Abox* $\mathcal{A}$. The legal KBs vary according to the DL used to express terminological and assertional axioms, and to the restrictions imposed on those axioms.

In this paper, we focus on DL-lite KBs.

3.1 DL-lite KBs

In DL-lite, the concepts and roles that can be built from atomic concepts and atomic roles are of the following form:
$B \rightarrow A \mid \exists R, \quad C \rightarrow B \mid \neg B, \quad R \rightarrow P \mid P^-, \quad E \rightarrow R \mid \neg R$
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).

The (set) semantics of concepts and roles is given in terms of interpretations. 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 non-atomic roles is defined as follows:

- $(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$,
- $(\exists R)^I = \{o_1 \mid \exists o_2 \ (o_1, o_2) \in R^I\}$, and
- $(\neg B)^I = \Delta^I \backslash B^I$ and $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$.

The axioms allowed in a Tbox of DL-lite[2] are concept inclusion constraints of the form $B \sqsubseteq C$, role inclusion constraints of the form $R \sqsubseteq E$, and functionality constraints on roles of the form $(funct\ R)$. It is important to note that general concepts or roles (i.e., with negation) are only allowed on the right hand side of inclusion constraints whereas only basic concepts or roles (i.e., without negation) occur on the left hand side of such constraints. Moreover, only basic roles occur in functionality constraints.

Inclusions of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$ are called *positive inclusions (PIs)*, while 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)*. PIs allow expressing inclusion dependencies, while NIs and functionalities allow expressing integrity constraints (ICs). For instance, in Figure 1, the PI $\exists$`hasTitle` $\sqsubseteq$ `Publication` expresses that only publications have titles, the NI `ConfPaper` $\sqsubseteq \neg$`JournPaper` expresses that conference proceeding papers and journal papers must be disjoint, and the functionality $(funct$ `hasTitle`$)$ expresses the usual (relational) functional dependency from the first attribute of `hasTitle` to the second one.

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 functionality constraint* $(funct\ R)$ if the binary relation R^I is a function, i.e., $(o, o_1) \in R^I$ and $(o, o_2) \in R^I$ implies $o_1 = o_2$. I is a *model of a Tbox* if it is a model of all of its constraints. A Tbox is *satisfiable* if it has a model. A Tbox $\mathcal{T}$ *logically entails* (a.k.a. *implies*) a constraint α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ is a model of α. For instance, in Figure 1, the PI `JournPaper` $\sqsubseteq \exists$`hasAuthor` is implied by $\mathcal{O}$ due to the presence of `JournPaper` $\sqsubseteq$ `Publication` and `Publication` $\sqsubseteq \exists$`hasAuthor` in $\mathcal{O}$. Finally, a Tbox $\mathcal{T}$ *logically entails* (a.k.a. *implies*) a Tbox $\mathcal{T}'$, written $\mathcal{T} \models \mathcal{T}'$, if every model of $\mathcal{T}$ is a model of $\mathcal{T}'$; and two Tboxes $\mathcal{T}$ and $\mathcal{T}'$ are *logically equivalent*, written $\mathcal{T} \equiv \mathcal{T}'$, iff $\mathcal{T} \models \mathcal{T}'$ and $\mathcal{T}' \models \mathcal{T}$.

An Abox consists of a finite set of membership assertions of the form $A(a)$ and $P(a, b)$, i.e., on atomic concepts and on atomic roles, 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.

An interpretation I is a *model of a KB* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable* (a.k.a. *consistent*) if it has at least one model. A KB $\mathcal{K}$ *logically entails* (a.k.a. implies) a constraint or assertion β, written $\mathcal{K} \models \beta$, if every model of $\mathcal{K}$ is a model of β. For instance, in Figure 1, the KB $\langle \mathcal{O}, \mathcal{D} \rangle$ of the DMS implies `JournPaper`(doi_2) due to `Survey` $\sqsubseteq$ `JournPaper` in $\mathcal{O}$ and `Survey`(doi_2) in $\mathcal{D}$.

Observe that any KB can be written equivalently as a first order logic (FOL) KB and a relational database following the *open-world assumption (OWA)*. The correspondences for Tbox constraints are summarized in Figure 4 for PIs, in Figure 5 for NIs, and in Figure 6 for functionalities. As for Abox assertions, they are simply FOL facts (i.e., ground atoms) and instances for atomic concepts and roles. Remark that, in constrast with usual databases following the *closed-world assumption (CWA)*, databases following OWA have incompletely specified data and their schemas (i.e., their constraints) are used to infer implicit facts, in addition to detect data inconsistencies. For example, we have seen above that $\langle \mathcal{O}, \mathcal{D} \rangle \models$ `JournPaper`(doi_2), while doi_2 is not explicitly stored in $\mathcal{D}$ in Figure 1. That is, doi_2 is an implicit instance of `JournPaper` in $\mathcal{D}$. This peculiarity has an important impact on most of the classical data management tasks like consistency checking and query answering [13].

3.2 Queries over a KB

A query q is of the form $q(\bar{x})$:- $\exists \bar{y} \ \phi(\bar{x}, \bar{y})$ where $\phi(\bar{x}, \bar{y})$ is a FOL formula, the variables of which are *only* the free variables $\bar{x}$ and the bound 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

2. These axioms actually correspond to the dialect DL-lite$_{\mathcal{FR}}$ of DL-lite.

DL notation	FOL notation	Relational notation (OWA)
$A \sqsubseteq A'$	$\forall X[A(X) \Rightarrow A'(X)]$	$A \subseteq A'$
$A \sqsubseteq \exists P$	$\forall X[A(X) \Rightarrow \exists Y P(X,Y)]$	$A \subseteq \Pi_1(P)$
$A \sqsubseteq \exists P^-$	$\forall X[A(X) \Rightarrow \exists Y P(Y,X)]$	$A \subseteq \Pi_2(P)$
$\exists P \sqsubseteq A$	$\forall X[\exists Y P(X,Y) \Rightarrow A(X)]$	$\Pi_1(P) \subseteq A$
$\exists P^- \sqsubseteq A$	$\forall X[\exists Y P(Y,X) \Rightarrow A(X)]$	$\Pi_2(P) \subseteq A$
$\exists Q \sqsubseteq \exists P$	$\forall X[\exists Y Q(X,Y) \Rightarrow \exists Z P(X,Z)]$	$\Pi_1(Q) \subseteq \Pi_1(P)$
$\exists Q \sqsubseteq \exists P^-$	$\forall X[\exists Y Q(X,Y) \Rightarrow \exists Z P(Z,X)]$	$\Pi_1(Q) \subseteq \Pi_2(P)$
$\exists Q^- \sqsubseteq \exists P$	$\forall X[\exists Y Q(Y,X) \Rightarrow \exists Z P(X,Z)]$	$\Pi_2(Q) \subseteq \Pi_1(P)$
$\exists Q^- \sqsubseteq \exists P^-$	$\forall X[\exists Y Q(Y,X) \Rightarrow \exists Z P(Z,X)]$	$\Pi_2(Q) \subseteq \Pi_2(P)$
$P \sqsubseteq Q^-$ or $P^- \sqsubseteq Q$	$\forall X,Y[P(X,Y) \Rightarrow Q(Y,X)]$	$P \subseteq \Pi_{2,1}(Q)$ or $\Pi_{2,1}(P) \subseteq Q$
$P \sqsubseteq Q$ or $P^- \sqsubseteq Q^-$	$\forall X,Y[P(X,Y) \Rightarrow Q(X,Y)]$	$P \subseteq Q$ or $\Pi_{2,1}(P) \subseteq \Pi_{2,1}(Q)$

Fig. 4. DL-lite PI axioms in FOL and relational notations. For the relational notation, which corresponds to unary and binary *inclusion dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively.

DL notation	FOL notation	Relational notation (OWA)
$A \sqsubseteq \neg A'$	$\forall X[A(X) \Rightarrow \neg A'(X)]$	$A \cap A' \subseteq \bot$
$A \sqsubseteq \neg\exists P$	$\forall X[A(X) \Rightarrow \neg\exists Y P(X,Y)]$	$A \cap \Pi_1(P) \subseteq \bot$
$A \sqsubseteq \neg\exists P^-$	$\forall X[A(X) \Rightarrow \neg\exists Y P(Y,X)]$	$A \cap \Pi_2(P) \subseteq \bot$
$\exists P \sqsubseteq \neg A$	$\forall X[\exists Y P(X,Y) \Rightarrow \neg A(X)]$	$A \cap \Pi_1(P) \subseteq \bot$
$\exists P^- \sqsubseteq \neg A$	$\forall X[\exists Y P(Y,X) \Rightarrow \neg A(X)]$	$A \cap \Pi_2(P) \subseteq \bot$
$\exists Q \sqsubseteq \neg\exists P$	$\forall X[\exists Y Q(X,Y) \Rightarrow \neg\exists Z P(X,Z)]$	$\Pi_1(Q) \cap \Pi_1(P) \subseteq \bot$
$\exists Q \sqsubseteq \neg\exists P^-$	$\forall X[\exists Y Q(X,Y) \Rightarrow \neg\exists Z P(Z,X)]$	$\Pi_1(Q) \cap \Pi_2(P) \subseteq \bot$
$\exists Q^- \sqsubseteq \neg\exists P$	$\forall X[\exists Y Q(Y,X) \Rightarrow \neg\exists Z P(X,Z)]$	$\Pi_2(Q) \cap \Pi_1(P) \subseteq \bot$
$\exists Q^- \sqsubseteq \neg\exists P^-$	$\forall X[Q(Y,X) \Rightarrow \neg\exists Z P(Z,X)]$	$\Pi_2(Q) \cap \Pi_2(P) \subseteq \bot$
$P \sqsubseteq \neg Q^-$ or $P^- \sqsubseteq \neg Q$	$\forall X,Y[P(X,Y) \Rightarrow \neg Q(Y,X)]$	$P \cap \Pi_{2,1}(Q) \subseteq \bot$ or $\Pi_{2,1}(P) \cap Q \subseteq \bot$
$P \sqsubseteq \neg Q$ or $P^- \sqsubseteq \neg Q^-$	$\forall X,Y[P(X,Y) \Rightarrow \neg Q(X,Y)]$	$P \cap Q \subseteq \bot$ or $\Pi_{2,1}(P) \cap \Pi_{2,1}(Q) \subseteq \bot$

Fig. 5. DL-lite NI axioms in FOL and relational notations. For the relational notation, which corresponds to *exclusion/disjointness dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively. We also assume that $\bot$ the empty relation.

DL notation	FOL notation	Relational notation (OWA)
$(funct\ P)$	$\forall X,Y,Z[P(X,Y) \wedge P(X,Z) \Rightarrow Y = Z]$	$P:\ 1 \rightarrow 2$
$(funct\ P^-)$	$\forall X,Y,Z[P(Y,X) \wedge P(Z,X) \Rightarrow Y = Z]$	$P:\ 2 \rightarrow 1$

Fig. 6. DL-lite functionality axioms in FOL and relational notations. For the relational notation, which corresponds to *functional dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively.

variables, e.g., 0 for a *boolean query*. When $\phi(\bar{x}, \bar{y})$ is of the form $conj(\bar{x}, \bar{y})$ where $conj(\bar{x}, \bar{y})$ is a conjunction of atoms, q is called a *conjunctive query*. Conjunctive queries, a.k.a. select-project-join queries, are the core database queries.

Given an interpretation $I = (\Delta^I, .^I)$, the semantics q^I of a boolean query q is defined as true if $[\phi(\emptyset, \bar{y})]^I = \mathrm{true}$, 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: $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\phi(\bar{e}, \bar{y})]^I = \mathrm{true}\}$. An interpretation that evaluates a boolean query to true, respectively a non-boolean query to a non empty set, is a *model* of that query.

3.3 Answer set of a query over a KB

Let q be a query over a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

If q is non-boolean, the answer set of q over $\mathcal{K}$ 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 constants appearing in the KB, $q(\bar{t})$ is the closed formula obtained by replacing in the query definition the free variables in $\bar{x}$ by the constants in $\bar{t}$, and $\mathcal{K} \models q(\bar{t})$ means as usual that every model of $\mathcal{K}$ is a model of $q(\bar{t})$.

If q is boolean, the answer set of q over $\mathcal{K}$ is by convention either $\{true\}$ or $\{false\}$: $ans(q, \mathcal{K}) = \{true\}$ if and only if $\mathcal{K} \models q()$, i.e., every model of $\mathcal{K}$ is a model of $q()$.

This corresponds to the so-called *certain answers semantics* requiring that an answer to a query, given a set of constraints (expressed here as a Tbox), to be an answer in all the models satisfying the constraints.

3.4 FOL-reducibility of data management

The DL-Lite family [11] has been designed so that data management is *FOL-reducible*. This property allows reducing a data management task over a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ to the evaluation against $\mathcal{A}$ *only* of a FOL query computed using $\mathcal{T}$ *only*.

The main idea of FOL-reducibility is to be able to perform a data management task in two separate steps: a first reasoning step that produces the FOL query and a second step which evaluates that query in a pure relational fashion. Indeed, FOL queries can be processed by SQL engines, thus taking advantage of well-established query optimization strategies supported by standard relational database management systems.

In fact, FOL-reducibility of data management holds in DL-lite only if we *forbid* functionality

constraints on roles involved in right-hand sides of role inclusion constraints[3]. For instance, in Figure 3, having $\texttt{hasContactAuthor} \sqsubseteq \texttt{hasAuthor}$ and $(funct\ \texttt{hasContactAuthor})$ in $\mathcal{O}''$ is legal, while having $\texttt{hasContactAuthor} \sqsubseteq \texttt{hasAuthor}$ and $(funct\ \texttt{hasAuthor})$ would be illegal. In the following, we only consider DL-lite Tboxes and KBs in which this restriction holds. Note that, as shown in [11], if we do not impose the above restriction on DL-lite, instance checking (a particular case of quey answering) is P-complete in data complexity, rulling out FOL-reducibility of query answering since data complexity of answering a FOL query is in $AC_0 \subset P$ [13], [14].

3.4.1 Consistency

It has been shown in [11] that given a Tbox $\mathcal{T}$, it is always possible to construct a FOL query q_{unsat} such that $ans(q_{unsat}, \mathcal{A}) = \{\text{true}\}$[4] iff the KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent, for any Abox $\mathcal{A}$ associated with $\mathcal{T}$.

Building the q_{unsat} query relies on the computation of the *IC-closure* of $\mathcal{T}$, i.e., the set of the integrity constraints (NI or functionality constraints) that are implied by $\mathcal{T}$: each constraint in the IC-closure is transformed into a conjunctive boolean query looking for counter-examples to it ; q_{unsat} is the union of these $unsat$ queries. The transformation of the integrity constraints into $unsat$ queries corresponds in fact to their negation and is summarized in Figure 7 for NIs and in Figure 8 for functionalities. For instance, the $unsat$ query corresponding to the negation of the NI $\texttt{JournPaper} \sqsubseteq \neg\texttt{ConfPaper}$ is $q()$:- $\exists x \texttt{JournPaper}(x) \wedge \texttt{ConfPaper}(x)$.

3.4.2 Query answering

It has been shown in [11] that given a Tbox $\mathcal{T}$ and for any query q built upon atomic concepts and roles of $\mathcal{T}$, it is always possible to construct a FOL query q_{rew} (called its *perfect rewriting*) such that $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle) = ans(q_{rew}, \mathcal{A})$ for any Abox $\mathcal{A}$ associated with $\mathcal{T}$.

Notably, [11] provides the $\mathsf{PerfectRef}(q, \mathcal{T})$ algorithm which computes the perfect rewriting q_{rew} of q using – PIs of – $\mathcal{T}$ only (i.e., independently of $\mathcal{A}$), which is a union of conjunctive queries built upon atomic concepts and roles of $\mathcal{T}$.

4 MODULE-BASED DATA MANAGEMENT

The main idea underlying the notion of module of a Tbox is to capture *some* constraints of the Tbox, including *all* the (implied) constraints built upon a given signature, denoted the *signature of interest*.

Our definition of *module* extends and encompasses the existing definitions. In contrast with [3], [4], [7], [9], we do not impose modules of a Tbox to be subsets of it. For a module to capture *some* constraints of the Tbox, it is indeed sufficient to impose that it is logically entailed by the Tbox. In contrast with [5], [6], [8], [10], we do not impose the signature of modules to be restricted to the signature of interest. In fact, as we have shown through the illustrative example, the robustness properties may enforce the signature of modules to contain additional relations that are not relations of interest but that are logically related to them.

Definition 1 (Module): Let $\mathcal{T}$ be a Tbox and $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$ a signature of interest. A *module* of $\mathcal{T}$ w.r.t. Γ is a Tbox $\mathcal{T}_\Gamma$ such that:

- $\Gamma \subseteq \mathsf{sig}(\mathcal{T}_\Gamma) \subseteq \mathsf{sig}(\mathcal{T})$,
- $\mathcal{T} \models \mathcal{T}_\Gamma$, and
- for any Tbox constraint α built upon Γ, $\mathcal{T} \models \alpha$ iff $\mathcal{T}_\Gamma \models \alpha$.

Notations. For distinguishing the relations of interest in the signature of a module $\mathcal{T}_\Gamma$ from those possibly imported from the reference Tbox for robustness purposes, we use the following notations: $\mathtt{r}$ denotes a relation of interest (i.e., in Γ), while $\mathtt{r}_{\text{ref}}$ denotes a relation of the reference Tbox.

We denote $\mathsf{sig}^+(\mathcal{T}_\Gamma)$ the set difference between the signature of $\mathcal{T}_\Gamma$ and Γ, i.e., the set of relations $\mathtt{r}^1_{\text{ref}}, \ldots, \mathtt{r}^k_{\text{ref}}$ of the reference Tbox that are involved in constraints of the module $\mathcal{T}_\Gamma$. Later on, we will denote $\mathtt{r}_{\text{mod}}$ the novel relations that may be added to the signature of a module for personalization purposes.

Example (continued) Consider the reference Tbox $\mathcal{O}$ of the running example. Consider the signature of interest $\Gamma = \{\texttt{JournPaper}, \texttt{HasAuthor}\}$. Let $\mathcal{T}^1_\Gamma$ and $\mathcal{T}^2_\Gamma$ be the following Tboxes: $\mathcal{T}^1_\Gamma = \{\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}\}$ and $\mathcal{T}^2_\Gamma = \{\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}, \texttt{JournPaper} \sqsubseteq \neg\texttt{ConfPaper}_{\text{ref}}\}$.

$\mathcal{T}^1_\Gamma$ and $\mathcal{T}^2_\Gamma$ are both modules of $\mathcal{O}$ w.r.t. Γ: they use relations from $\mathcal{O}$ only ; without being subsets of $\mathcal{O}$, they are implied by $\mathcal{O}$ ($\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$ is implied by $\texttt{JournPaper} \sqsubseteq \texttt{Publication}_{\text{ref}}$ and $\texttt{Publication}_{\text{ref}} \sqsubseteq \exists\texttt{hasAuthor}$ in $\mathcal{O}$, and $\texttt{JournPaper} \sqsubseteq \neg\texttt{ConfPaper}_{\text{ref}}$ is equivalent to $\texttt{ConfPaper}_{\text{ref}} \sqsubseteq \neg\texttt{JournPaper}$ in $\mathcal{O}$) ; the only constraint built upon Γ that is implied by $\mathcal{O}$ is $\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$, which is in both $\mathcal{T}^1_\Gamma$ and $\mathcal{T}^2_\Gamma$ (and thus implied by them).

Here, $\mathsf{sig}^+(\mathcal{T}^1_\Gamma) = \emptyset$ since $\mathcal{T}^1_\Gamma$ involves only relations of interest (i.e., in Γ), while $\mathsf{sig}^+(\mathcal{T}^2_\Gamma) = \{\texttt{ConfPaper}_{\text{ref}}\}$. ■

It is worth noticing that, as the above example shows, a module of a Tbox w.r.t. a signature of interest may not be unique.

4.1 Robust module-based data management

We define now the two notions of *robustness* for modules that have been illustrated in Section 2.

Notations. From now on, $\mathcal{A}_{/\mathsf{sig}}$ denotes the restriction of an Abox $\mathcal{A}$ to the assertions of $\mathcal{A}$ built upon the signature sig only.

3. This corresponds to the dialect DL-lite$_\mathcal{A}$ of DL-lite.

4. By a slight abuse of notation, we denote hereinafter the answer set $ans(q, \langle \emptyset, \mathcal{A} \rangle)$ of a query q over a KB $\langle \emptyset, \mathcal{A} \rangle$ by $ans(q, \mathcal{A})$, which corresponds exactly to the standard relational evaluation of q against the relational database $\mathcal{A}$.

NI	Corresponding $unsat$ query
$A \sqsubseteq \neg A'$ or $A' \sqsubseteq \neg A$	$\exists X[A(X) \wedge A'(X)]$
$A \sqsubseteq \neg\exists P$ or $\exists P \sqsubseteq \neg A$	$\exists X, Y[A(X) \wedge P(X, Y)]$
$A \sqsubseteq \neg\exists P^-$ or $\exists P^- \sqsubseteq \neg A$	$\exists X, Y[A(X) \wedge P(Y, X)]$
$\exists Q \sqsubseteq \neg\exists P$ or $\exists P \sqsubseteq \neg\exists Q$	$\exists X, Y, Z[Q(X, Y) \wedge P(X, Z)]$
$\exists Q \sqsubseteq \neg\exists P^-$ or $\exists P^- \sqsubseteq \neg\exists Q$	$\exists X, Y, Z[Q(X, Y) \wedge P(Z, X)]$
$\exists Q^- \sqsubseteq \neg\exists P$ or $\exists P \sqsubseteq \neg\exists Q^-$	$\exists X, Y, Z[Q(Y, X) \wedge P(X, Z)]$
$\exists Q^- \sqsubseteq \neg\exists P^-$	$\exists X, Y, Z[Q(Y, X) \wedge P(Z, X)]$
$P \sqsubseteq \neg Q^-$ or $Q^- \sqsubseteq \neg P$ or $P^- \sqsubseteq \neg Q$ or $Q \sqsubseteq P^-$	$\exists X, Y[P(X, Y) \wedge Q(Y, X)]$
$P \sqsubseteq \neg Q$ or $Q \sqsubseteq \neg P$ or $P^- \sqsubseteq \neg Q^-$ or $Q^- \sqsubseteq \neg P^-$	$\exists X, Y[P(X, Y) \wedge Q(X, Y)]$

Fig. 7. From NI axioms to $unsat$ queries.

Functionality	Corresponding $unsat$ query
$(funct\ P)$	$\exists X, Y, Z[P(X, Y) \wedge P(X, Z) \wedge Y \neq Z)]$
$(funct\ P^-)$	$\exists X, Y, Z[P(Y, X) \wedge P(Z, X) \wedge Y \neq Z)]$

Fig. 8. From functionality axioms to $unsat$ queries.

4.1.1 Robustness to consistency checking

A module of a Tbox $\mathcal{T}$ w.r.t. a signature Γ is *robust to consistency checking* (Definition 2) if it contains enough constraints to detect data inconsistency due to Γ.

Definition 2 (Robustness to consistency checking): Let $\mathcal{T}_\Gamma$ be a module of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \text{sig}(\mathcal{T})$. $\mathcal{T}_\Gamma$ is *robust to consistency checking* iff for every Abox $\mathcal{A}_\Gamma$ built upon Γ and for every Abox $\mathcal{A}_{\bar{\Gamma}}$ built upon $\text{sig}(\mathcal{T})\backslash\Gamma$ that is consistent with $\mathcal{T}$: $\langle \mathcal{T}, \mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}} \rangle$ is consistent iff $\langle \mathcal{T}_\Gamma, (\mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}})_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent.

Example (continued) Consider again the above two modules $\mathcal{T}_\Gamma^1$ and $\mathcal{T}_\Gamma^2$ of $\mathcal{O}$ w.r.t. $\Gamma = \{\texttt{JournPaper}, \texttt{hasAuthor}\}$.

Neither $\mathcal{T}_\Gamma^1$ nor $\mathcal{T}_\Gamma^2$ is robust to consistency checking, as shown by the following counter-example. Let $\mathcal{T}_\Gamma$ be either $\mathcal{T}_\Gamma^1$ or $\mathcal{T}_\Gamma^2$, $\mathcal{A}_\Gamma$ be $\{\texttt{Publication}(a)\}$, and $\mathcal{A}_{\bar{\Gamma}}$ be $\{\texttt{FullPaper}_{\text{ref}}(a)\}$. $\mathcal{A}_{\bar{\Gamma}/\text{sig}(\mathcal{T}_\Gamma)}$ is empty, thus $(\mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}})_{/\text{sig}(\mathcal{T}_\Gamma)} = \{\texttt{Publication}(a)\}$, so $\langle \mathcal{T}_\Gamma, (\mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}})_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent. However, $\langle \mathcal{O}, \mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}} \rangle$ is inconsistent: $\texttt{FullPaper}_{\text{ref}} \sqsubseteq \neg\texttt{JournPaper}$ is implied by $\texttt{FullPaper}_{\text{ref}} \sqsubseteq \texttt{ConfPaper}_{\text{ref}}$ and $\texttt{ConfPaper}_{\text{ref}} \sqsubseteq \neg\texttt{JournPaper}$ in $\mathcal{O}$, and violated by $(\mathcal{A}_\Gamma \cup \mathcal{A}_{\bar{\Gamma}}) = \{\texttt{JournPaper}(a), \texttt{FullPaper}_{\text{ref}}(a)\}$.

The algorithms and results of Section 5 will allow checking that the following $\mathcal{T}_\Gamma^3$ is robust to consistency checking:

$\mathcal{T}_\Gamma^3 = \{\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor},$
$\texttt{JournPaper} \sqsubseteq \neg\texttt{ConfPaper}_{\text{ref}},$
$\texttt{JournPaper} \sqsubseteq \neg\texttt{FullPaper}_{\text{ref}},$
$\texttt{JournPaper} \sqsubseteq \neg\texttt{ShortPaper}_{\text{ref}},$
$\texttt{Survey}_{\text{ref}} \sqsubseteq \texttt{JournPaper}\}.$ ■

Theorem 1 shows that global consistency (see Definition 3) of a module-based DMS built from a module robust to consistency checking can be verified by restricting the construction of the $unsat$ queries to the integrity constraints (ICs) implied by the module, and then by evaluating their union against the dataset distributed among the module-based DMS and the reference one.

Definition 3 (Global consistency): Let $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}', \mathcal{A}' \rangle$ be two consistent KBs. $\langle \mathcal{T}', \mathcal{A}' \rangle$ is *globally consistent w.r.t.* $\langle \mathcal{T}, \mathcal{A} \rangle$ iff $\langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent.

In the following, we consider the global consistency of a module-based KB $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ w.r.t. a reference KB $\langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}_\Gamma$ is a module of $\mathcal{T}$. From now on, if it is clear from the context, we omit to mention the reference KB when we refer to global consistency.

Theorem 1 (Global consistency checking): Let $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ be consistent KBs such that $\mathcal{T}_\Gamma$ is a module robust to consistency checking of $\mathcal{T}$ w.r.t. a signature Γ, and $\mathcal{A}'$ is an Abox built upon Γ. Let $q_{\text{unsat}}(\mathcal{T}_\Gamma)$ be the union of $unsat$ queries obtained from the IC-closure of $\mathcal{T}_\Gamma$. Then: $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ is globally consistent iff $ans(q_{\text{unsat}}(\mathcal{T}_\Gamma), \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}') = \{false\}$.

Proof: We first show, based on the standard first order semantics of DL-lite, that: $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent iff $\langle \mathcal{T}_\Gamma, \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle$ is consistent. For one direction, this follows from: $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle \models \langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle$ with $(\mathcal{A} \cup \mathcal{A}')_{/\text{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$. For the converse direction, $\langle \mathcal{T}_\Gamma, \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle$ is consistent, so $\langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle$ is also consistent, since $(\mathcal{A} \cup \mathcal{A}')_{/\text{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$. Now, consider the subset $(\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}}$ of $\mathcal{A} \cup \mathcal{A}'$ built upon $\text{sig}(\mathcal{T})\backslash\Gamma$, and its complement $(\mathcal{A} \cup \mathcal{A}')_\Gamma$. We therefore get: $\langle \mathcal{T}_\Gamma, ((\mathcal{A} \cup \mathcal{A}')_\Gamma \cup (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}})_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle$ is consistent. By construction, $(\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \subseteq \mathcal{A}$. Since the KB $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent and $\langle \mathcal{T}, \mathcal{A} \rangle \models \langle \mathcal{T}, (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \rangle$, $\langle \mathcal{T}, (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \rangle$ is also consistent. As a result, Definition 2 applies and we get: $\langle \mathcal{T}, (\mathcal{A} \cup \mathcal{A}')_\Gamma \cup (\mathcal{A} \cup \mathcal{A}')_{\bar{\Gamma}} \rangle$ is consistent, i.e., $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent.

Theorem 1 then follows from the two-step process for checking consistency in a DL-lite KB $\langle \mathcal{T}, \mathcal{A} \rangle$: computation of the IC-closure of the Tbox $\mathcal{T}$ and translation of each IC in the IC-closure into a conjunctive $unsat$ query; the KB is consistent iff the union q_{unsat} of all the $unsat$ queries is evaluated to $false$ against $\mathcal{A}$ seen as a relational database. Since checking the global consistency (i.e., whether $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}' \rangle$ is consistent) is equivalent to checking the consistency of $\langle \mathcal{T}_\Gamma, \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}' \rangle$, as shown above, it can be done by evaluating the union of $unsat$ queries obtained from the IC-closure of $\mathcal{T}_\Gamma$ (instead of $\mathcal{T}$) against the distributed dataset $\mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$ (instead of $\mathcal{A} \cup \mathcal{A}'$). □

Corollary 1 provides a simple way to optimize global consistency checking by evaluating a *necessary and sufficient subset* of the unsat queries obtained from the IC-closure of a module robust to consistency checking. It directly follows from the fact that in Theorem 1 each KB

$\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ taken in isolation is consistent. That is, the only (implied) ICs that can be violated are those requiring the *two* Aboxes in order to be checked.

Corollary 1: Let $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ be consistent KBs such that $\mathcal{T}_\Gamma$ is a module robust to consistency checking of $\mathcal{T}$ w.r.t. a signature Γ, and $\mathcal{A}'$ is an Abox built upon Γ. Then: $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ is globally consistent iff $ans(q^{\text{opt}}_{\text{unsat}}(\mathcal{T}_\Gamma), \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}') = \{false\}$, where $q^{\text{opt}}_{\text{unsat}}(\mathcal{T}_\Gamma)$ is the union of the $unsat$ queries obtained from the IC-closure of $\mathcal{T}_\Gamma$, *without* considering the integrity constraints involving relations from $sig^+(\mathcal{T}_\Gamma)$ only (i.e., of the form $\mathtt{r}_{\text{ref}}$).

Example (continued) The NI $\mathtt{Survey}_{\text{ref}} \sqsubseteq \neg \mathtt{FullPaper}_{\text{ref}}$ is in the IC-closure of the module $\mathcal{T}^3_\Gamma$: it is implied by $\mathtt{Survey}_{\text{ref}} \sqsubseteq \mathtt{JournPaper}$ and $\mathtt{JournPaper} \sqsubseteq \neg \mathtt{ConfPaper}_{\text{ref}}$ in $\mathcal{T}^3_\Gamma$. However, the corresponding $unsat$ query is useless to check global consistency (as it would check the consistency of the reference DMS which is known to be consistent), and can be discarded from the $unsat$ queries to evaluate. ■

A global inconsistency may arise from an update of the reference DMS or of the module-based DMS. Using the above results, such a global inconsistency can be checked *on demand* or *on a periodical basis*.

In addition, global consistency can also be checked upon each module-based DMS's update. Corollary 2 provides a way to simplify the $unsat$ queries to be posed to the reference DMS, in function of the update.

In the following, we assume w.l.o.g. that an update adds or modifies a *single* tuple in the module-based DMS. The generalization to an update involving several tuples is obvious. Observe that deletions are not considered here since they cannot lead to a global inconsistency. Indeed, DL-lite, as a FOL language, is monotonic.

Definition 4 (Unsat query relevant to update): Let $\mathcal{T}_\Gamma$ be a module defined w.r.t. a signature Γ. Let A and P be an atomic concept and an atomic role in Γ respectively, and $A'_{(\text{ref})}$ and $Q_{(\text{ref})}$ respectively be an atomic concept and an atomic role in either Γ or $sig^+(\mathcal{T}_\Gamma)$.

An unsat query is *relevant to an update* $A(a)$ iff it is of one of the following forms:

- $q()$:- $A'_{(\text{ref})}(a)$ s.t. either $A \sqsubseteq \neg A'_{(\text{ref})}$ or $A'_{(\text{ref})} \sqsubseteq \neg A$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ Q_{(\text{ref})}(a, Y)$ s.t. either $A \sqsubseteq \neg \exists Q_{(\text{ref})}$ or $\exists Q_{(\text{ref})} \sqsubseteq \neg A$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ Q_{(\text{ref})}(Y, a)$ s.t. either $A \sqsubseteq \neg \exists Q^-_{(\text{ref})}$ or $\exists Q^-_{(\text{ref})} \sqsubseteq \neg A$ is in the IC-closure of $\mathcal{T}_\Gamma$.

An unsat query is *relevant to an update* $P(a, b)$ iff it is of one of the following forms:

- $q()$:- $Q_{(\text{ref})}(a, b)$ s.t. either $P \sqsubseteq \neg Q_{(\text{ref})}$ or $Q_{(\text{ref})} \sqsubseteq \neg P$, or $P^- \sqsubseteq \neg Q^-_{(\text{ref})}$ or $Q^-_{(\text{ref})} \sqsubseteq \neg P^-$, is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $Q_{(\text{ref})}(b, a)$ s.t. either $P \sqsubseteq \neg Q^-_{(\text{ref})}$ or $Q^-_{(\text{ref})} \sqsubseteq \neg P$, or $P^- \sqsubseteq \neg Q_{(\text{ref})}$ or $Q_{(\text{ref})} \sqsubseteq \neg P^-$, is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ Q_{(\text{ref})}(a, Y)$ s.t. either $\exists P \sqsubseteq \neg \exists Q_{(\text{ref})}$ or $\exists Q_{(\text{ref})} \sqsubseteq \neg \exists P$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ Q_{(\text{ref})}(Y, a)$ s.t. either $\exists P \sqsubseteq \neg \exists Q^-_{(\text{ref})}$ or $\exists Q^-_{(\text{ref})} \sqsubseteq \neg \exists P$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ Q_{(\text{ref})}(Y, b)$ s.t. either $\exists P^- \sqsubseteq \neg \exists Q^-_{(\text{ref})}$ or $\exists Q^-_{(\text{ref})} \sqsubseteq \neg \exists P^-$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y \ R_{(\text{ref})}(b, Y)$ s.t. either $\exists P^- \sqsubseteq \neg \exists Q_{(\text{ref})}$ or $\exists Q_{(\text{ref})} \sqsubseteq \neg \exists P^-$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y P(a, Y) \wedge Y \neq b$ s.t. $(funct \ P)$ is in the IC-closure of $\mathcal{T}_\Gamma$,
- $q()$:- $\exists Y P(Y, b) \wedge Y \neq a$ s.t. $(funct \ P^-)$ is in the IC-closure of $\mathcal{T}_\Gamma$.

Corollary 2 (Global consistency upon local update): Let $\langle \mathcal{T}, \mathcal{A} \rangle$ and $\langle \mathcal{T}_\Gamma, \mathcal{A}' \rangle$ be consistent KBs such that $\mathcal{T}_\Gamma$ is a module robust to consistency checking of $\mathcal{T}$ w.r.t. a signature Γ, and $\mathcal{A}'$ is an Abox built upon Γ. Let μ be an update of $\mathcal{A}'$ and $\mu(\mathcal{A}')$ its result. Let $q_{\text{unsat}}(\mu, \mathcal{T}_\Gamma)$ be the union of $unsat$ queries *relevant to* μ. Then: $\langle \mathcal{T}_\Gamma, \mu(\mathcal{A}') \rangle$ is globally consistent iff $ans(q_{\text{unsat}}(\mu, \mathcal{T}_\Gamma), \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)}) = \{false\}$.

Example (continued) Consider the update that amounts to adding $\mathtt{JournPaper}(doi_1)$ to a module-based DMS built from the module $\mathcal{T}^3_\Gamma$. Checking the global consistency upon this update can be done by asking the union of the $unsat$ queries built from the IC-closure of $\mathcal{T}^3_\Gamma$ that are *relevant* to the update: $q_1()$:- $\mathtt{ConfPaper}_{\text{ref}}(doi_1)$, $q_2()$:- $\mathtt{FullPaper}_{\text{ref}}(doi_1)$, and $q_3()$:- $\mathtt{ShortPaper}_{\text{ref}}(doi_1)$. ■

4.1.2 Robustness to query answering

A module of a Tbox $\mathcal{T}$ defined w.r.t. a signature Γ is *robust to query answering* if it contains enough knowledge to compute the answer set of any query built upon Γ asked to any Abox associated with $\mathcal{T}$.

Definition 5 (Robustness to query answering): Let $\mathcal{T}_\Gamma$ be a module of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \text{sig}(\mathcal{T})$. $\mathcal{T}_\Gamma$ is *robust to query answering* iff for every Abox $\mathcal{A}$ consistent with $\mathcal{T}$ and for every query q built upon Γ: $ans(q, \langle \mathcal{T}, \mathcal{A} \rangle) = ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\text{sig}(\mathcal{T}_\Gamma)} \rangle)$.

Example (continued) The algorithms and results of Section 5 will enable to check that the following module is robust to query answering:

$\mathcal{T}^4_\Gamma = \{\mathtt{JournPaper} \sqsubseteq \exists \mathtt{hasAuthor},$
$\mathtt{Publication}_{\text{ref}} \sqsubseteq \exists \mathtt{hasTitle}_{\text{ref}},$
$\mathtt{Publication}_{\text{ref}} \sqsubseteq \exists \mathtt{hasAuthor},$
$\exists \mathtt{hasTitle}_{\text{ref}} \sqsubseteq \mathtt{Publication}_{\text{ref}},$
$\mathtt{ConfPaper}_{\text{ref}} \sqsubseteq \mathtt{Publication}_{\text{ref}},$
$\mathtt{JournPaper} \sqsubseteq \mathtt{Publication}_{\text{ref}},$
$\mathtt{ShortPaper}_{\text{ref}} \sqsubseteq \mathtt{ConfPaper}_{\text{ref}},$
$\mathtt{FullPaper}_{\text{ref}} \sqsubseteq \mathtt{ConfPaper}_{\text{ref}},$
$\mathtt{Survey}_{\text{ref}} \sqsubseteq \mathtt{JournPaper}\}$.

However, the module $\mathcal{T}^3_\Gamma$ is not robust to query answering, as shown by the following counter-example. Consider the Abox $\mathcal{A} = \{\mathtt{ConfPaper}_{\text{ref}}(a)\}$ for the reference Tbox $\mathcal{O}$ and the query $q(x)$:- $\exists y \ \mathtt{hasAuthor}(x, y)$ built upon Γ. $a \in ans(q, \langle \mathcal{T}, \mathcal{A} \rangle)$ because $\mathtt{ConfPaper}_{\text{ref}} \sqsubseteq \exists \mathtt{hasAuthor}$ is implied by $\mathcal{O}$, but $ans(q, \langle \mathcal{T}^3_\Gamma, \mathcal{A}_{/\text{sig}(\mathcal{T}^3_\Gamma)} \rangle) = \emptyset$ since, although $\mathtt{ConfPaper}_{\text{ref}}(a) \in \mathcal{A}_{/\text{sig}(\mathcal{T}^3_\Gamma)}$, $\mathtt{ConfPaper}_{\text{ref}} \sqsubseteq \exists \mathtt{hasAuthor}$ is not implied by $\mathcal{T}^3_\Gamma$. ■

Theorem 2 shows that if a module-based DMS is built from a module robust to query answering, global query answering (see Definition 6) can be done by computing the perfect rewriting of the query using the module only, and then by evaluating it against the dataset distributed among the module-based DMS and the reference one.

Definition 6 (Global answer set): Let $\langle \mathcal{T}, \mathcal{A}\rangle$ and $\langle \mathcal{T}', \mathcal{A}'\rangle$ be consistent KBs, and q a query over $\langle \mathcal{T}', \mathcal{A}'\rangle$. The *global answer set* of q w.r.t. $\langle \mathcal{T}, \mathcal{A}\rangle$, denoted $global\_ans(q, \langle \mathcal{T}, \mathcal{A}\rangle, \langle \mathcal{T}', \mathcal{A}'\rangle)$, is: $ans(q, \langle \mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}'\rangle)$. In the following, we consider the global answer sets of queries over a module-based KB $\langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle$ and w.r.t. a reference KB $\langle \mathcal{T}, \mathcal{A}\rangle$ where $\mathcal{T}_\Gamma$ is a module of $\mathcal{T}$. From now on, if it is clear from the context, we omit to mention the reference KB when we refer to global answer sets.

Theorem 2 (Global query answering): Let $\langle \mathcal{T}, \mathcal{A}\rangle$ and $\langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle$ be KBs such that $\mathcal{T}_\Gamma$ is a module of $\mathcal{T}$ w.r.t. Γ which is robust to query answering, and $\mathcal{A}'$ is built upon Γ. Let q be a conjunctive query built upon Γ and $q_{rew}(\mathcal{T}_\Gamma)$ its perfect rewriting w.r.t. $\mathcal{T}_\Gamma$. If $\langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle$ is globally consistent, then: $global\_ans(q, \langle \mathcal{T}, \mathcal{A}\rangle, \langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle) = ans(q_{rew}(\mathcal{T}_\Gamma), \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}')$.

Proof: By definition, $global\_ans(q, \langle \mathcal{T}, \mathcal{A}\rangle, \langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle) = ans(q, \langle \mathcal{T} \cup \mathcal{T}_\Gamma, \mathcal{A} \cup \mathcal{A}'\rangle) = ans(q, \langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}'\rangle)$ since $\mathcal{T} \models \mathcal{T}_\Gamma$. From the global consistency of $\langle \mathcal{T}_\Gamma, \mathcal{A}'\rangle$, we get the consistency of $\langle \mathcal{T} \cup \mathcal{T}_\Gamma, \mathcal{A} \cup \mathcal{A}'\rangle$ from Definition 3, i.e., of $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}'\rangle$. Definition 5 now applies and we get: $ans(q, \langle \mathcal{T}, \mathcal{A} \cup \mathcal{A}'\rangle) = ans(q, \langle \mathcal{T}_\Gamma, (\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)}\rangle) = ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'\rangle)$, since $(\mathcal{A} \cup \mathcal{A}')_{/\mathsf{sig}(\mathcal{T}_\Gamma)} = \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$.

Theorem 2 then follows from the fact that in DL-lite $ans(q, \langle \mathcal{T}_\Gamma, \mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'\rangle)$ can be obtained by evaluating the perfect rewriting $q_{rew}(\mathcal{T}_\Gamma)$ of q w.r.t. $\mathcal{T}_\Gamma$ against the distributed dataset $\mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$. □

Observe that Theorem 2 applies when a module-based DMS is globally consistent. From a practical viewpoint, this means that a module robust to query answering should also be robust to consistency checking, so that global consistency can be effectively checked by a module-based DMS. In general, Theorem 2 does not hold when a module-based DMS is globally inconsistent. The reason is that any tuple – of the arity of a query – built from the Aboxes of the reference and module-based DMSs (i.e., $\mathcal{A} \cup \mathcal{A}'$) is then an answer, while it may not be possible to exhibit every such tuple at the module-based DMS level which only has a partial view of those two Aboxes (i.e., $\mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma)} \cup \mathcal{A}'$).

Example (continued) Consider again the module $\mathcal{T}_\Gamma^4$ of $\mathcal{O}$ w.r.t. $\Gamma = \{\texttt{JournPaper}, \texttt{hasAuthor}\}$ that is robust to query answering. Let the dataset $\mathcal{D}$ in Figure 1 be the Abox $\mathcal{A}$ associated with $\mathcal{O}$. Let $\mathcal{A}' = \{\texttt{JournPaper}(doi_1)\}$ be the Abox associated with $\mathcal{T}_\Gamma^4$. $\langle \mathcal{T}_\Gamma^4, \mathcal{A}'\rangle$ is globally inconsistent since doi_1 cannot be at the same time both a journal paper and a (full) paper in conference proceedings. As a result, "1998" (like any other constant in $\mathcal{A}$) is a global answer to the query $q(x)$:- $\texttt{JournPaper}(x)$. However, that answer cannot be obtained from, as "1998" is not in, $\mathcal{A}_{/\mathsf{sig}(\mathcal{T}_\Gamma^4)} \cup \mathcal{A}'$. ■

4.2 Safe personalization in module-based data management

We investigate now how the *personalization* of a module – through updates – can preserve its global data management skills. Indeed, it is often necessary to personalize an extracted module so that it fully copes with the new application needs.

Definition 7 (Personalization of a module): Let $\mathcal{T}$ be a Tbox and let $\mathcal{T}_\Gamma$ be a module of $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. A Tbox $\mathcal{T}'$ is a *personalization* of $\mathcal{T}_\Gamma$ iff $\mathcal{T}'$ is obtained from $\mathcal{T}_\Gamma$ after a sequence of insertions or deletions of Tbox constraints (i.e., PIs, NIs, or functionalities).

Definitions 8 and 9 exhibit *safeness* conditions both at the Tbox and the Abox levels of a module-based DMS. Theorems 3 and 4 then show that these conditions are *sufficient* to perform global data management of any *safe instance* of a *safe personalization of a module*.

A personalization of a module is safe provided that: first, the updates cannot involve atomic concepts and roles of the reference DMS's Tbox other than those in the signature of the module; second, the updates must conform with the reference DMS's Tbox; third, the resulting updated Tbox must be a module of the reference DMS's Tbox with the same robustness(es) as the personalized module.

Definition 8 (Safe personalization of a module): Let $\mathcal{T}_\Gamma$ be a module of a Tbox $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. A personalization $\mathcal{T}'$ of $\mathcal{T}_\Gamma$ is *safe* w.r.t. $\mathcal{T}$ iff

1) $\mathsf{sig}(\mathcal{T}) \cap (\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}(\mathcal{T}_\Gamma)) = \emptyset$,
2) $\mathcal{T}$ is a module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T})$, and
3) $\mathcal{T}'$ is a module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma)$ with the same robustness(es) as $\mathcal{T}_\Gamma$.

Example (continued) Consider the personalization $\mathcal{T}'$ of the module $\mathcal{T}_\Gamma^4$ obtained by adding the constraints (thus the novel atomic role $\texttt{hasRightOn}_{\text{mod}}$): $\texttt{hasAuthor} \sqsubseteq \texttt{hasRightOn}^-_{\text{mod}}$ and $\exists \texttt{hasRightOn}^-_{\text{mod}} \sqsubseteq \texttt{JournPaper}$, stating that authors have rights on their journal papers.

$\mathcal{T}'$ is not a safe personalization of $\mathcal{T}_\Gamma^4$ w.r.t. $\mathcal{O}$ because $\mathcal{O}$ is not a module of $\mathcal{O} \cup \mathcal{T}'$: $\exists \texttt{hasAuthor} \sqsubseteq \texttt{JournPaper}$ (which is built upon the signature of $\mathcal{O}$) is indeed implied by $\mathcal{O} \cup \mathcal{T}'$ – actually by the above two constraints – but not by $\mathcal{O}$ alone, thus does not conform with $\mathcal{O}$.

In contrast, the algorithms and results of Section 5 will enable to check that $\mathcal{T}''$, obtained by adding to $\mathcal{T}_\Gamma^4$ the constraint $\texttt{hasAuthor} \sqsubseteq \texttt{hasRightOn}^-_{\text{mod}}$, is a safe personalization of $\mathcal{T}_\Gamma^4$ w.r.t. $\mathcal{O}$. ■

A safe instance of a personalization $\mathcal{T}'$ of a module $\mathcal{T}_\Gamma$ is an Abox built *only* upon the relations of Γ that remain in $\mathcal{T}'$ after personalization of $\mathcal{T}_\Gamma$, plus the novel relations introduced in $\mathcal{T}'$ by personalization of $\mathcal{T}_\Gamma$.

Definition 9 (Safe instance): Let $\mathcal{T}_\Gamma$ be a module of a Tbox $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. Let $\mathcal{T}'$ be a personalization of $\mathcal{T}_\Gamma$. A safe instance of $\mathcal{T}'$ is an Abox built upon $\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma)$, where $\mathsf{sig}^+(\mathcal{T}_\Gamma)$ is the set difference between the signature of $\mathcal{T}_\Gamma$ and Γ.

Example (continued) Consider the above safe personalization $\mathcal{T}''$ of $\mathcal{T}^4_\Gamma$ w.r.t. $\mathcal{O}$.

The Abox $\mathcal{A}_1$ is an instance of $\mathcal{T}''$ that is not safe because it states information on the relation $\texttt{Publication}_{\text{ref}}$ which is in $\text{sig}^+(\mathcal{T}^4_\Gamma)$: $\mathcal{A}_1 = \{\texttt{Publication}_{\text{ref}}(a), \texttt{hasContactAuthor}_{\text{mod}}(a,c)\}$.

In contrast, the Abox $\mathcal{A}_2$ is a safe instance of $\mathcal{T}'$: $\mathcal{A}_2 = \{\texttt{JournPaper}(a), \texttt{hasContactAuthor}_{\text{mod}}(a,c)\}$. ■

Theorems 3 and 4 result directly from Definitions 8 and 9 by adapting accordingly the proofs of Theorems 1 and 2.

Theorem 3 (Safe global consistency checking): Let $\langle \mathcal{T}, \mathcal{A}\rangle$ and $\langle \mathcal{T}', \mathcal{A}'\rangle$ be consistent KBs, such that $\mathcal{T}'$ is a safe personalization of a module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ, $\mathcal{T}_\Gamma$ being robust to consistency checking, and $\mathcal{A}'$ is a safe instance of $\mathcal{T}'$. Let $q_{\text{unsat}}(\mathcal{T}')$ be the union of unsat queries obtained from the IC-closure of $\mathcal{T}'$. Then: $\langle \mathcal{T}', \mathcal{A}'\rangle$ is globally consistent iff $ans(q_{\text{unsat}}(\mathcal{T}'), \mathcal{A}_{/\text{sig}(\mathcal{T}')} \cup \mathcal{A}') = \{false\}$.

Theorem 4 (Safe global query answering): Let $\langle \mathcal{T}, \mathcal{A}\rangle$ and $\langle \mathcal{T}', \mathcal{A}'\rangle$ be KBs, such that $\mathcal{T}'$ is a safe personalization of a module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ, $\mathcal{T}_\Gamma$ being robust to query answering, and $\mathcal{A}'$ is a safe instance of $\mathcal{T}'$. Let q be a query built upon $\text{sig}(\mathcal{T}')\backslash\text{sig}^+(\mathcal{T}_\Gamma)$ and $q_{rew}(\mathcal{T}')$ its perfect rewriting w.r.t. $\mathcal{T}'$. If $\langle \mathcal{T}', \mathcal{A}'\rangle$ is globally consistent, then: $global\_ans(q, \langle \mathcal{T}, \mathcal{A}\rangle, \langle \mathcal{T}', \mathcal{A}'\rangle) = ans(q_{rew}(\mathcal{T}'), \mathcal{A}_{/\text{sig}(\mathcal{T}')} \cup \mathcal{A}')$.

Remark 1: Similarly to Corollary 1, safe global consistency checking can be optimized by evaluating the $unsat$ queries obtained from a necessary and sufficient subset of the IC-closure of the personalized module: those that are not built upon relations of the reference DMS only (i.e., of the form $\texttt{r}_{\text{ref}}$), and those built only upon novel relations resulting from personalization (i.e., of the form $\texttt{r}_{\text{mod}}$).

Remark 2: Corollary 2 can also be adapted to *safe global consistency upon update*, provided a slight extension of the definition of an $unsat$ query relevant to an update (Definition 4): updates now concern both atomic concepts and roles of the form A or A_{mod}, and P or P_{mod} respectively; and unsat queries relevant to an update are defined from the minimal subset of ICs for checking safe global consistency (Remark 1).

4.3 Optimizing practical module-based data management using minimal modules

In general, several (possibly personalized) modules of a Tbox may exist for a same signature of interest. Notably, a module always exists since a Tbox is a module of itself (robust to both consistency checking and query answering).

To compare the existing modules for a given Tbox and a given signature, we define *minimal* modules based on the notions of *syntactic* minimality and of *semantic* minimality. Syntactic minimality deals with redundancy within a module, while semantic minimality deals with the amount of useless extra-knowledge captured within a module w.r.t. the given signature and expected robustness(es).

Definition 10 (Minimal module): Let $\mathcal{T}'$ be a (safely personalized) module of a Tbox $\mathcal{T}$ w.r.t. Γ, possibly robust to query answering and/or consistency checking.

- $\mathcal{T}'$ is *syntactically minimal* iff any strict subset of $\mathcal{T}'$ is not equivalent to $\mathcal{T}'$.
- $\mathcal{T}'$ is *semantically minimal* iff for any (safely and identically personalized) module $\mathcal{T}''$ of $\mathcal{T}$ w.r.t. Γ with the same robustness(es): if $\mathcal{T}' \models \mathcal{T}''$ then $\mathcal{T}'' \equiv \mathcal{T}'$.
- $\mathcal{T}'$ is *minimal* iff it is both syntactically and semantically minimal.

It is worth noticing that minimal modules are desirable, since non-minimality induces useless extra-computation in practical module-based DMSs (e.g., QuOnto[5]), as illustrated below.

Example (continued) Consider again the module $\mathcal{T}^3_\Gamma$ that is robust to consistency checking. $\mathcal{T}^3_\Gamma$ is not – *semantically* – minimal, as the constraint $\texttt{Survey}_{\text{ref}} \sqsubseteq \texttt{JournPaper}$ is neither required by a module w.r.t. Γ nor by its robustness to consistency checking. The issue is that at (global) consistency checking time, such a constraint is used – by state of the art algorithms (e.g., [11]) – to compute the IC-closure of the module, thus may lead to the generation of useless ICs and to the construction and evaluation of the corresponding unsat queries. Here, the useless generated constraints would be: $\texttt{Survey}_{\text{ref}} \sqsubseteq \neg\texttt{ConfPaper}_{\text{ref}}$, $\texttt{Survey}_{\text{ref}} \sqsubseteq \neg\texttt{FullPaper}_{\text{ref}}$, and $\texttt{Survey}_{\text{ref}} \sqsubseteq \neg\texttt{ShortPaper}_{\text{ref}}$.

Now, consider again the module $\mathcal{T}^4_\Gamma$ that is robust to query answering. $\mathcal{T}^4_\Gamma$ is not – *syntactically* – minimal, as the constraint $\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$ is redundant: it is implied by $\texttt{JournPaper} \sqsubseteq \texttt{Publication}_{\text{ref}}$ and $\texttt{Publication}_{\text{ref}} \sqsubseteq \exists\texttt{hasAuthor}$ in $\mathcal{T}^4_\Gamma$. The issue is that at (global) query answering time, such a constraint is used – by state of the art algorithms (e.g., [11]) – to compute the perfect rewriting of a query, leading to useless computation. For intance, consider the query $q(x)$:- $\exists y\ \texttt{hasAuthor}(x,y)$. From q and $\texttt{Publication}_{\text{ref}} \sqsubseteq \exists\texttt{hasAuthor}$, $\text{PerfectRef}(q, \mathcal{T}^4_\Gamma)$ first produces $q'(x)$:- $\texttt{Publication}_{\text{ref}}(x)$ from which in turn it produces $q''(x)$:- $\texttt{JournPaper}(x)$ using $\texttt{JournPaper} \sqsubseteq \texttt{Publication}_{\text{ref}}$. This reformulation is redundant since from q and $\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$, $\text{PerfectRef}(q, \mathcal{T}^4_\Gamma)$ also produces $q''(x)$:- $\texttt{JournPaper}(x)$. ■

5 Algorithms for Robust Module-Based Data Management

We provide here algorithms for extracting modules from a DL-lite Tbox (Section 5.1), checking safe module personalization (Section 5.2) – checking whether an instance is safe is trivial –, and minimizing a module (Section 5.3).

5.1 Extracting robust modules

The ERM algorithm for extracting robust modules from a given DL-lite Tbox (Algorithm 1) relies on the notion of (deductive) closure of a Tbox.

5. http://www.dis.uniroma1.it/quonto/

Definition 11 (Closure of a Tbox): The *closure* of a Tbox $\mathcal{T}$ is the Tbox $cl(\mathcal{T})$ made of every constraint (PI, NI, and functionality) implied by $\mathcal{T}$ (i.e., $cl(\mathcal{T}) = \{\alpha \mid \mathcal{T} \models \alpha\}$).

The following theorem characterizes the main properties of the closure of a Tbox that will be used later on.

Theorem 5 (Properties of Tbox closure): Let $\mathcal{T}$ be a DL-lite Tbox.

1) The size of $cl(\mathcal{T})$ is at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$.
2) Computing $cl(\mathcal{T})$ is polynomial in the size of $\mathsf{sig}(\mathcal{T})$.
3) $\mathcal{T}$ and $cl(\mathcal{T})$ are equivalent and for any constraint α built upon $\mathsf{sig}(\mathcal{T})$: $\mathcal{T} \models \alpha$ iff $\alpha \in cl(\mathcal{T})$.

Proof: The first item follows from the fact that the number of atomic concepts, respectively of atomic roles, is bounded by the size $\#\mathsf{sig}(\mathcal{T})$ of $\mathsf{sig}(\mathcal{T})$. It follows that the number of possible $B \sqsubseteq C$ in $\mathcal{T}$ is bounded by $18 \times \#\mathsf{sig}(\mathcal{T})^2$ since there are $3 \times \#\mathsf{sig}(\mathcal{T})$ possible B's and $6 \times \#\mathsf{sig}(\mathcal{T})$ possible C's, as a B is of the form A, $\exists P$, or $\exists P^-$, while a C is of the form B or $\neg B$; the number of possible $R \sqsubseteq E$ in $\mathcal{T}$ is bounded by $8 \times \#\mathsf{sig}(\mathcal{T})^2$ since there are $2 \times \#\mathsf{sig}(\mathcal{T})$ possible R's and $4 \times \#\mathsf{sig}(\mathcal{T})$ possible E's, as a R is of the form P or P^-, while a E is of the form R or $\neg R$; the number of possible $(funct\ R)$ is bounded by $2 \times \#\mathsf{sig}(\mathcal{T})$ since there are $2 \times \#\mathsf{sig}(\mathcal{T})$ possible R's as mentioned above. As a result, the size of $cl(\mathcal{T})$ is at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$.

The second item follows from the fact that checking whether a Tbox $\mathcal{T}$ implies a constraint α is polynomial in the size of $\mathcal{T}$ in DL-lite (Theorem 27 in [11]), this size being bounded by that of its closure (first item).

Finally, the third item trivially follows from Definition 11: $\mathcal{T} \models cl(\mathcal{T})$ and $\mathcal{T} \subseteq cl(\mathcal{T})$, thus $cl(\mathcal{T}) \models \mathcal{T}$. □

With the definition of closure in place, we provide the Extract Robust Module (ERM) algorithm (Algorithm 1) that builds a module of a Tbox w.r.t. a signature and the desired robustness(es). The resulting module is semantically minimal and is obtained by filtering the closure of the Tbox or the Tbox itself for keeping the relevant constraints.

Theorem 6 (Properties of the ERM *algorithm):* Let $\mathcal{T}$ be a DL-lite Tbox and Γ a subset of $\mathsf{sig}(\mathcal{T})$. $\mathsf{ERM}(\mathcal{T}, \Gamma, RQA, RCC)$ terminates and returns a module of $\mathcal{T}$ w.r.t. Γ, which is semantically minimal, robust to query answering if $RQA = \text{true}$, and robust to consistency checking if $RCC = \text{true}$.

Proof: Termination for lines 2–6 follows from the finiteness of $cl(\mathcal{T})$, and for lines 7–13 because in the worst case $\mathcal{T} \subseteq \mathcal{T}_\Gamma$, so $\mathcal{T}_\Gamma = \mathcal{T}'_\Gamma$ eventually.

$\mathcal{T}_\Gamma$, the output of $\mathsf{ERM}(\mathcal{T}, \Gamma, RQA, RCC)$, is a module of Tbox $\mathcal{T}$ w.r.t. Γ (Definition 1) since $\mathcal{T}_\Gamma \subseteq cl(\mathcal{T})$ and the filtering condition of line 3.

If $RCC = \text{true}$, robustness to consistency checking is guaranteed by the lines 5–6, which precisely extract all (and *only*) the implied ICs exhibited in Corollary 1.

If $RQA = \text{true}$, robustness to query answering is guaranteed by the lines 7–13, which precisely extract all (and *only*) the PIs required for constructing the perfect rewriting of any conjunctive query q built upon Γ, i.e., those used by $\mathsf{PerfectRef}(q, \mathcal{T})$ (Lemma 39 in [11]).

Finally, if $RCC = \text{false}$ and $RQA = \text{false}$, $\mathcal{T}_\Gamma$ is semantically minimal since it is constructed from lines 1–4, thus built upon Γ only. If $RCC = \text{true}$ or $RQA = \text{true}$, minimality results from the fact that *only* the required constraints are added by lines 5–6 and 7–13. □

Algorithm 1: the ERM algorithm
$\mathsf{ERM}(\mathcal{T}, \Gamma, RQA, RCC)$
Input: a DL-lite Tbox $\mathcal{T}$, a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, two booleans RQA and RCC
Output: a module $\mathcal{T}_\Gamma$ of $\mathcal{T}$ w.r.t. Γ, which is semantically minimal, robust to query answering if $RQA = \text{true}$, and robust to consistency checking if $RCC = \text{true}$
(1) $\mathcal{T}_\Gamma \leftarrow \emptyset$
(2) **foreach** $\alpha \in cl(\mathcal{T})$
(3) **if** α is built upon Γ only
(4) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{\alpha\}$
(5) **else if** $RCC = \text{true}$ and α is a NI $X \sqsubseteq \neg Y$ s.t. X or Y is built upon Γ
(6) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{\alpha\}$
(7) **if** $RQA = \text{true}$
(8) $\mathsf{sig} \leftarrow \Gamma$; $\mathcal{T}'_\Gamma \leftarrow \emptyset$
(9) **while** $\mathcal{T}_\Gamma \neq \mathcal{T}'_\Gamma$
(10) $\mathcal{T}'_\Gamma \leftarrow \mathcal{T}_\Gamma$; $\mathsf{sig}' \leftarrow \mathsf{sig}$
(11) **foreach** PI $X \sqsubseteq Y \in \mathcal{T}$ s.t. Y is built upon sig'
(12) $\mathcal{T}_\Gamma \leftarrow \mathcal{T}_\Gamma \cup \{X \sqsubseteq Y\}$
(13) $\mathsf{sig} \leftarrow \mathsf{sig} \cup \mathsf{sig}_X$ (for X built upon the signature sig_X)
(14) **return** $\mathcal{T}_\Gamma$

Corollary 3 states the complexity of computing (robust) modules.

Corollary 3 (Complexity of module extraction): Let $\mathcal{T}$ be a DL-lite Tbox and $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. Computing a module of $\mathcal{T}$ w.r.t. Γ, possibly robust to query answering and/or consistency checking, is polynomial in the size of $\mathsf{sig}(\mathcal{T})$.

Proof: The worst computational cost is that of a module robust to both consistency checking and query answering. $\mathsf{ERM}(\mathcal{T}, \Gamma, \text{true}, \text{true})$ first computes a module robust to consistency checking using lines 1–6. The closure of $\mathcal{T}$ must be computed and then its constraints are filtered. This can be done in polynomial time in the size of $\mathsf{sig}(\mathcal{T})$ according to Theorem 5. Then, $\mathsf{ERM}(\mathcal{T}, \Gamma, \text{true}, \text{true})$ filters $\mathcal{T}$ for robustness to query answering. In the worst case, according to Theorem 5, we have a number of iterations of the **while** loop that is at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$ (since $\mathcal{T}_\Gamma \subseteq cl(\mathcal{T})$), each of which has to filter $\mathcal{T}$ whose size is at most quadratic in the size of $\mathsf{sig}(\mathcal{T})$. As a result, the worst case computation of a module of $\mathcal{T}$ w.r.t. Γ is polynomial in the size of $\mathsf{sig}(\mathcal{T})$. □

5.2 Checking safe personalization of a module

The Safe Personalization Checking (SPC) algorithm (Algorithm 2) checks whether the personalization $\mathcal{T}'$ of a module $\mathcal{T}_\Gamma$ a of Tbox $\mathcal{T}$ w.r.t. a signature Γ is safe. It

Algorithm 2: The SPC algorithm
$\mathsf{SPC}(\mathcal{T}', \mathcal{T}_\Gamma, \mathcal{T}, RQA, RCC)$
Input: a Tbox $\mathcal{T}'$ that is a personalization of the module $\mathcal{T}_\Gamma$ of a Tbox $\mathcal{T}$ w.r.t. $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, and two booleans RQA and RCC denoting respectively whether $\mathcal{T}_\Gamma$ is robust to query answering and/or consistency checking
Output: true if $\mathcal{T}'$ is safe, false otherwise
(1) **if** $\mathsf{sig}(\mathcal{T}) \cap (\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}(\mathcal{T}_\Gamma)) \neq \emptyset$
(2) **return** false
(3) **if** $cl(\mathcal{T}) \neq \mathsf{ERM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}), \text{false}, \text{false})$
(4) **return** false
(5) **if** $cl(\mathsf{ERM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma), RQA, RCC)) \neq cl(\mathcal{T}')$
(6) **return** false
(7) **return** true

proceeds by checking sequentially whether one of the three conditions of Definition 8 is violated.

Theorem 7 (Properties of the SPC *algorithm):* Let $\mathcal{T}$ be a Tbox and $\mathcal{T}_\Gamma$ a module of $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$, the robustness of which is specified by two booleans RQA and RCC ($RQA = \text{true}$ iff it is robust to query answering, $RCC = \text{true}$ iff it is robust to consistency checking). Let $\mathcal{T}'$ be a personalization of $\mathcal{T}_\Gamma$: $\mathsf{SPC}(\mathcal{T}', \mathcal{T}_\Gamma, \mathcal{T}, RQA, RCC)$ terminates and returns true iff $\mathcal{T}'$ is a safe personalization of $\mathcal{T}_\Gamma$ w.r.t. $\mathcal{T}$.

Proof: Termination directly follows from the finiteness of the closure of a Tbox and from the termination of the ERM algorithm (Theorem 6).

As for correctness, line 1 of Algorithm 2 obviously checks whether the first item 1 of Definition 8 is falsified. For checking whether $\mathcal{T}$ is not a module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T})$, the second item of Definition 8, based on the properties of the closure of a Tbox (item *3.* of Theorem 5), line 3 of Algorithm 2 compares the closure of $\mathcal{T}$ with the closure of a semantically minimal module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T})$. The latter is computed by $\mathsf{ERM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}), \text{false}, \text{false})$, as the result of the loop starting at Line 2 in Algorithm 1. Finally, for checking whether the third item of Definition 8 is falsified, line 5 of Algorithm 2 compares the closure of $\mathcal{T}'$ with the closure of a semantically minimal module of $\mathcal{T} \cup \mathcal{T}'$ w.r.t. $\mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma)$ with the same robustness(es) as $\mathcal{T}_\Gamma$. Because of the lines 7–13 of Algorithm 1 that work on $\mathcal{T}$ instead of $cl(\mathcal{T})$ like the previous lines, when $RQA = \text{true}$, it may not be the case that $\mathsf{ERM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma), RQA, RCC)$ outputs the closure of a module. Therefore, comparing the closure of $\mathcal{T}'$ with the closure of $\mathsf{ERM}(\mathcal{T} \cup \mathcal{T}', \mathsf{sig}(\mathcal{T}') \backslash \mathsf{sig}^+(\mathcal{T}_\Gamma), RQA, RCC)$ is necessary in line 5 of Algorithm 2 for checking the last item of Definition 8. □

Corollary 4 states the complexity of checking whether a personalization of a module is safe. It directly follows from the polynomial time and size of the closure construction (Theorem 5).

Corollary 4 (Complexity of checking safeness): Let $\mathcal{T}'$ be the personalization of a (possibly robust) module $\mathcal{T}_\Gamma$ of a Tbox $\mathcal{T}$ w.r.t. a signature $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. Deciding whether $\mathcal{T}'$ is a safe personalization of $\mathcal{T}_\Gamma$ w.r.t. $\mathcal{T}$ is polynomial in the size of $\mathsf{sig}(\mathcal{T} \cup \mathcal{T}')$.

5.3 Computing minimal modules by reduction

As mentioned in Section 4, minimal modules play an important role for the efficiency of practical module-based data management.

We build minimal modules by computing syntactically minimal modules from (possibly personalized) semantically minimal modules. To do so, we introduce the notion of *reduction* of a Tbox for removing redundant constraints. This operation is dual to that of closure.

Definition 12 (Tbox Reduction): The *reduction* of a Tbox $\mathcal{T}$, denoted $red(\mathcal{T})$, is obtained by starting from $red(\mathcal{T}) = \mathcal{T}$, then applying exhaustively the following rule until no more constraints can be removed from $red(\mathcal{T})$:
If $\alpha \in red(\mathcal{T})$ and $red(\mathcal{T}) \backslash \{\alpha\} \models \alpha$, then:
remove α from $red(\mathcal{T})$.

The following theorem characterizes the main properties of the reduction of a Tbox.

Theorem 8 (Properties of Tbox reduction): Let $\mathcal{T}$ be a Tbox.

1) Computing $red(\mathcal{T})$ is polynomial in the size of $\mathsf{sig}(\mathcal{T})$.
2) $\mathcal{T}$ and $red(\mathcal{T})$ are equivalent.
3) Any strict subset of $red(\mathcal{T})$ is not equivalent to $\mathcal{T}$.

Proof: The first item results from the fact that the size of a Tbox is bounded by that of its closure and checking whether a Tbox implies a constraint is polynomial in the size of that Tbox in DL-lite (Theorem 27 in [11]), that size being bounded by that of its closure. The last two items directly follow from Definition 12. □

Corollary 5 provides a simple means to compute (possibly personalized) minimal modules. It follows from Definition 8, Theorem 6 (which states that ERM outputs semantically minimal modules), and Theorem 8.

Corollary 5 (Computation/complexity of minimal modules): Let $\mathcal{T}$ be a Tbox and $\Gamma \subseteq \mathsf{sig}(\mathcal{T})$. Let $\mathcal{T}'$ be a safe personalization of (the reduction of) $\mathsf{ERM}(\mathcal{T}, \Gamma, RQA, RCC)$. $red(\mathcal{T}')$ is minimal and is computed in polynomial time in the size of $sig(\mathcal{T}')$.

6 Conclusion

The modules introduced in this paper generalize both the modules obtained by extracting a subset of a Tbox w.r.t. selected relations (e.g., [3], [4], [7], [9]) or by forgetting about relations (e.g., [5], [6], [8], [10]).

In addition, in contrast with existing work, we have considered the problem of safe personalization of modules built from an existing reference DMS. This raises new issues to check easily that a module-based DMS evolves independently but coherently w.r.t. the reference DMS from which it has been built. We have introduced two notions of module robustness that make possible to build locally the relevant queries to ask to the reference

database in order to check global consistency (possibly upon each update), and to obtain global answers for local queries. We have provided polynomial time algorithms that extract minimal and robust modules from a reference ontological schema expressed as a DL-lite Tbox.

[5] extracts modules from DL-lite schemas following a forgetting approach. It proposes an alternative to our result about global query answering, which applies under the severe constraints that the dataset of the reference DMS has to be modified (*write* access is required).

Compared to the algorithm developed by [7] for extracting modules from acyclic $\mathcal{EL}$ ontological schemas, our approach handles possibly *cyclic* DL-lite$_\mathcal{A}$ schemas, while keeping data consistency and query answering reducible to standard database queries [11].

In contrast with the recent work on extracting modules from DL-lite ontological schema [4], we focus on the DL-lite$_\mathcal{A}$ fragment for which consistency checking and query answering are FOL-reducible. This is crucial when ontologies are used as schemas over large datasets stored and queried as relational databases.

Datalog$^\pm$ [15] is an extension of Datalog that has also been designed for query answering over ontologies. Since it captures the fragment of DL-lite that we consider, our results can be easily transposed into it.

Contrarily to recent works in distributed databases, data replication can be avoided while guaranteeing global consistency. Our approach is a good trade-off between the NoSQL approaches and the SQL approaches for managing distributed data stores (see [16] for a survey). While most of the NoSQL approaches are schemaless, our approach makes possible to handle useful schema constraints. It provides efficient means to check global consistency, a stronger property than eventual consistency that is prevalent in distributed data stores. On the other hand, we are more flexible than the SQL approaches since global consistency is checked periodically and not at each update of the reference DMS.

In the next future, we plan to evaluate our approach, in particular to compare the size of the modules extracted by our algorithm to the results provided by [17], [18]. We also plan to apply our algorithms to the real use-case of the MyCorporisFabrica DMS, mentioned in the introduction, which has been developed manually as a personalization of the (reference) Foundational Model of Anatomy DMS. Finally, we plan to extend our approach to distributed module-based DMSs, where answering queries combines knowledge of several modules associated with possibly several reference DMSs.

REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] H. Stuckenschmidt, C. Parent, and S. Spaccapietra, Eds., *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, ser. Lecture Notes in Computer Science. Springer, 2009, vol. 5445.

[3] S. Ghilardi, C. Lutz, and F. Wolter, "Did i damage my ontology? a case for conservative extensions in description logics," in *KR*, 2006.

[4] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev, "Minimal module extraction from DL-Lite ontologies using QBF solvers," in *IJCAI*, 2009.

[5] Z. Wang, K. Wang, R. W. Topor, and J. Z. Pan, "Forgetting concepts in DL-Lite," in *ESWC*, 2008.

[6] B. Konev, D. Walther, and F. Wolter, "Forgetting and uniform interpolation in extensions of the description logic EL," in *Description Logics*, 2009.

[7] B. Konev, C. Lutz, D. Walther, and F. Wolter, "Semantic modularity and module extraction in description logics," in *ECAI*, 2008.

[8] B. Konev, D. Walther, and F. Wolter, "Forgetting and uniform interpolation in large-scale description logic terminologies," in *IJCAI*, 2009.

[9] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "Just the right amount: extracting modules from ontologies," in *WWW*, 2007.

[10] K. Wang, Z. Wang, R. W. Topor, J. Z. Pan, and G. Antoniou, "Concept and role forgetting in ALC ontologies," in *ISWC*, 2009.

[11] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable reasoning and efficient query answering in description logics: The *dl-lite* family," *JAR*, vol. 39, no. 3, pp. 385–429, 2007.

[12] O. Palombi, G. Bousquet, D. Jospin, S. Hassan, L. Revéret, and F. Faure, "My corporis fabrica: A unified ontological, geometrical and mechanical view of human anatomy," in *3DPH*, 2009.

[13] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[14] M. Y. Vardi, "The complexity of relational query languages," in *STOC*, 1982.

[15] A.Cali, G.Gottlob, and T. Lukasiewicz, "Datalog+-: a unified approach to ontologies and integrity constraints," in *Proc. Intl. Conf. on Database Theory (ICDT)*, 2009.

[16] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Record*, vol. 39, no. 4, pp. 12–27, 2010.

[17] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "Extracting modules from ontologies: A logic-based approach," in *OWLED*, 2007.

[18] ——, "Modular reuse of ontologies: Theory and practice," *J. Artif. Intell. Res. (JAIR)*, vol. 31, pp. 273–318, 2008.

François Goasdoué is an Associate Professor of Computer Science at Université Paris-Sud, where he is member of the LRI (Laboratoire de Recherche en Informatique). His areas of research are Databases, Knowledge Representation & Reasoning, and the Semantic Web.

Marie-Christine Rousset is a Professor of Computer Science at the Université de Grenoble, where she is member of the LIG (Laboratoire d'Informatique de Grenoble). Her areas of research are Knowledge Representation & Reasoning, Information Integration, Pattern Mining, and the Semantic Web.