Towards Techniques for Updating Virtual Knowledge Graphs

Romuald Esdras Wandji[1,*], Mantas Šimkus[1] and Diego Calvanese[1,2]

[1]Department of Computing Science, Umeå Universitet, Umeå, Sweden
[2]Research Centre for Knowledge and Data, Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy

Abstract
The field of Virtual Knowledge Graphs (VKGs) continues to grow in both academic and applied contexts. Yet, the issue of updates in VKG systems has not yet received adequate attention, although it is crucial to manage data modifications at the data source level through the lens of an ontology. In this paper, we focus on VKGs whose ontology is specified in the lightweight ontology language *DL-Lite*$_A$, and we propose diverse settings and research directions we intend to explore to address the challenge of translating ontology-based updates into updates at the level of data sources. We also pay attention to the important problem of automated analysis of mappings, which plays a major role when it comes to reformulating ontology-based update requests into update requests over the data sources.

Keywords
Knowledge Representation, Virtual Knowledge Graph (VKG), Ontology-based Data Access, View Updates

1. Introduction

Virtual Knowledge Graphs (VKGs) is a rapidly growing field that aims to bridge the gap between the Semantic Web technologies and database systems providing a robust methodology for integrating heterogeneous data sources with the assistance of ontologies. An ontology layer in a VKG provides a unified abstract view of the application domain, allows for more expressive data querying, and improves data integration and interoperability [1, 2, 3, 4]. This paradigm features several characteristics that proved to be effective in managing complex information systems. The content of a (virtual) knowledge graph is generated from data sources through the use of declarative *mappings*. It is typically represented using a formal language such as OWL 2 (Web Ontology Language) or one of its profiles (i.e., sub-languages) [5], and in our framework, we are particularly interested in *DL-Lite*$_A$ ontologies [1]. As a member of the *DL-Lite* family of *Description Logics (DLs)* [6], *DL-Lite*$_A$ is characterized by its ability to capture conceptual modeling formalisms and by the fact that it enjoys efficient reasoning [7, 2]. Most importantly, *DL-Lite*$_A$ enjoys *first-order rewritability*, i.e., every (union of) conjunctive queries posed over

RuleML+RR'23: 17th International Rule Challenge and 7th Doctoral Consortium, September 18–20, 2023, Oslo, Norway
*Corresponding author.

romuald.esdras.wandji@umu.se (R. E. Wandji); mantas.simkus@umu.se (M. Šimkus); diego.calvanese@unibz.it (D. Calvanese)
https://www.umu.se/en/staff/romuald-esdras-wandji (R. E. Wandji)
0009-0008-5036-2452 (R. E. Wandji); 0000-0003-0632-0294 (M. Šimkus); 0000-0001-5174-9693 (D. Calvanese)

the ontology layer can be translated into an equivalent first-order query that can be executed over the data source layer [2].

The focus of research in VKGs has been on query answering, i.e., on using the ontology layer and the corresponding knowledge graph to extract data specified through a query from the underlying data sources. However, ontology-based updates in VKGs remains a challenging task, which refers to modifying its *extensional content* with new knowledge that may or may not be consistent w.r.t. the original one. This task comes with some important challenges. The first is due to the complex logic-based inferences in ontologies that lead to the need for a non-deterministic approach to updating knowledge bases [8, 9]. Another important challenge comes from the presence of mappings that are essentially considered views that define ontology elements in terms of queries over the data source. Thus, each update over an ontology element can be translated into an update over the view that combines all queries that correspond to mappings for that ontology element and thus faces the *view-update* problem [10, 11, 12]. Several methodologies have emerged within the realm of ontology-based updates, chiefly addressing two primary classes of updates as delineated in [9]. The first category is the *instance-level* (or ABox) updates, where an update consists of changing assertions about individual entities in the ontology. This is the most common form of updates studied in the literature [13, 14, 15, 16], originating from the work in [17]. Concurrently, an alternative update methodology is characterized by a *formula-based* approach [18, 19], which encapsulates updates as logical formulas, representing a semantic change that needs to be reflected in the ontology and is an example of the *Set-Of-Theories* approach [20]. However these two update approaches are limited to the ontology layer, i.e., they allow users to modify the *extensional content* of the ontology, but mappings and thus updates of actual data sources are not covered by these previous works.

The main focus of this PhD work is to address the fact that little attention has been paid so far to the problem of *updating* VKGs through operations that are performed at the ontology level, where changes applied over the ontology are translated into minimally suitable updates over the underlying database through the mappings. A solution to this problem would be of great practical relevance since it would allow the management of the key operations that are of interest in an information system (i.e., queries and updates) through the lens of ontology.

This article initially presents our motivation behind exploring the concept of ontology-based updates within the context of VKGs. To this end, we recall the notion of knowledge base updates as it is investigated in the literature, which provides us key definitions that will be useful for further investigation in our specific research context. Subsequently, we introduce the concept of updates in VKGs, where the goal is to manipulate data source instances by means of operations performed over the ontology layer. We outline the current research challenges for the study of VKG updates, which include defining an inconsistency-tolerant semantics for updates in VKGs, investigating computational properties of update and query operations under such semantics, and finding conditions under which VKG updates can bypass ontology axioms (via query rewriting). Furthermore, we want to study how to enrich a given VKG system with additional information to ensure a unique translation of updates at the ontological level to updates of the data sources.

2. Preliminaries

Here we briefly introduce standard DL *Knowledge Bases (KBs)*. Subsequently, we provide the definition of *DL-Lite*$_A$, the specific DL variant that forms the focus of this research study. Our exposition concludes with an explanation of the concept of VKGs and its principal components. In the following, we assume to have three distinct, countably infinite alphabets: N_O for ontology predicates, N_S for source predicates, and N_I for constants. These alphabets are pairwise disjoint.

2.1. Description Logic knowledge bases

Description Logics (DLs) [6] serve as the foundation of contemporary knowledge representation systems, designed to depict the universe of discourse in a logically consistent and computationally tractable manner. Acting as the building blocks of this domain, DLs employ two fundamental entities: *concepts* and *roles*. Concepts denote unary relations, whereas roles denote binary relations, and both are constructed by applying specific constructs starting from atomic concepts and roles.

DLs allow for modeling a domain of interest by employing concepts and roles within a *knowledge base* (KB). Specifically, a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ in a DL consists of two components: the *TBox* $\mathcal{T}$ and the *ABox* $\mathcal{A}$. A TBox is the intensional level of the KB, capturing the terminological knowledge (or schema) of the domain. An *ABox* comprises information at the instance level, providing concrete instances of the concepts and roles defined in the TBox.

In this paper, we center our attention on a specific family of DLs known as *DL-Lite* [7, 1], which aligns with the tractable OWL 2 QL profile [5] of the Web Ontology Language OWL 2, and, specifically, we consider *DL-Lite*$_A$, which is one of the most expressive variants in the *DL-Lite* family for which a query-rewriting based approach to query answering can be adopted. In *DL-Lite*$_A$, a TBox is defined as a finite set of intensional assertions of the form, $B_1 \sqsubseteq B_2$ (*concept inclusion*), $B_1 \sqsubseteq \neg B_2$ (*concept disjointness*), $R_1 \sqsubseteq R_2$ (*role inclusion*), $R_1 \sqsubseteq \neg R_2$ (*role disjointness*),and $(\mathsf{funct}\ R)$ (*functionality*). Here, R (possibly sub-scripted) denotes an atomic role P or its inverse P^-. Instead, B (possibly sub-scripted) denotes a *basic concept*, which is either an atomic concept A, or a concept defined as $\exists R$, representing the set of objects that appear as the first argument of R. Finally, $(\mathsf{funct}\ R)$ asserts the functionality of R, i.e., that its first argument operates as a key of the relation denoted by R. An ABox in *DL-Lite*$_A$ is represented as a finite set of assertions of the form $A(a)$ or $P(a, b)$, with a and b belonging to N_I.

The semantics of a DL KB is given in terms of First-Order interpretations, referring to the interpretation structures of First-Order Logic [6]. An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of an interpretation *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$, which maps each atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role P to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The role P^- is interpreted as the inverse of the relation $P^{\mathcal{I}}$, while $(\exists R)^{\mathcal{I}} = \{o \mid \text{there exists } o' \text{ s.t. } (o, o') \in R^{\mathcal{I}}\}$. Finally, $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$, and $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is said to be a *model* of (or *satisfies*) an ABox assertion $A(a)$ (resp., $P(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp., $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$), and it is a model of a TBox assertion $E_1 \sqsubseteq E_2$ (where E_1 and E_2 are either concepts or roles) if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of all assertions in $\mathcal{T}$ and $\mathcal{A}$. We denote with $Mod(\mathcal{K})$ the set of all models of $\mathcal{K}$.

A KB $\mathcal{K}$ is *consistent* if it has at least one model, i.e., $Mod(\mathcal{K}) \neq \emptyset$. We say that $\mathcal{A}$ is *consistent with* $\mathcal{T}$, or $\mathcal{T}$*-consistent*, if $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent, and $\mathcal{T}$*-inconsistent* otherwise. Given an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, we denote by $cl_{\mathcal{T}}(\mathcal{A})$ the *closure* of $\mathcal{A}$ w.r.t. $\mathcal{T}$ that is, the set of ABox assertions over individuals in $\mathcal{A}$ that are logically implied by $\langle \mathcal{T}, \mathcal{A} \rangle$. Similarly, the deductive closure of a TBox $\mathcal{T}$, denoted $cl(\mathcal{T})$ is the set of inclusions and functionality assertions that follow from $\mathcal{T}$. Finally, given two ABoxes $\mathcal{A}$ and $\mathcal{A}'$, we say that $\mathcal{A}$ is logically equivalent to $\mathcal{A}'$ w.r.t. $\mathcal{T}$, denoted $\mathcal{A} \equiv_{\mathcal{T}} \mathcal{A}'$ if $cl_{\mathcal{T}}(\mathcal{A}) = cl_{\mathcal{T}}(\mathcal{A}')$.

2.2. Virtual Knowledge Graphs

We recall here the main elements of the VKG framework, which is also known as *Ontology-based Data Access* (OBDA) [3]. A *VKG specification* is a tuple $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, where *(i)* $\mathcal{T}$ is a TBox (expressed in OWL 2 QL), *(ii)* $\mathcal{S}$ is a (possibly federated) data source schema, and *(iii)* $\mathcal{M}$ is a set of global-as-view (GAV) mapping assertions [21], each of which specifies how to populate a concept or a role of the ontology in terms of a query over the data source. Specifically, each mapping assertion has the form $\varphi(\vec{x}) \rightsquigarrow \psi(\vec{x})$ where $\varphi(\vec{x})$ is a first-order query (typically written in SQL syntax) over the predicates in N_S of the data source $\mathcal{S}$, and $\psi(\vec{x})$ is a conjunction of atoms over $\mathcal{T}$.

A *VKG instance* is a pair $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$ where $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ is a *VKG specification*, and $\mathcal{D}$ is a relational database instance compliant with $\mathcal{S}$. Its semantics is defined in terms of FO interpretations over $\mathcal{T}$. An interpretation $\mathcal{I}$ is a *model* of $\langle \mathcal{P}, \mathcal{D} \rangle$ if *(i)* $\mathcal{I} \models \mathcal{T}$ and *(ii)* $\mathcal{I}$ satisfies the set $\mathcal{M}(\mathcal{D})$ of facts retrieved through $\mathcal{M}$ from $\mathcal{D}$, where $\mathcal{M}(\mathcal{D})$ is formally defined as follows:

$$\mathcal{M}(\mathcal{D}) = \{\psi(\vec{t}) \mid \vec{t} \in eval(\varphi(\vec{x}), \mathcal{D}) \text{ and } \varphi(\vec{x}) \rightsquigarrow \psi(\vec{x}) \in \mathcal{M}\},$$

where $eval(\varphi(\vec{x}), \mathcal{D})$ is the evaluation of $\varphi(\vec{x})$ over $\mathcal{D}$. In other words, an interpretation $\mathcal{I}$ is a model of $\langle \mathcal{P}, \mathcal{D} \rangle$ if it is a model of $\langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle$. It is worth noting that the retrieved ABox $\mathcal{M}(\mathcal{D})$ is kept virtual, which means that for tasks such as query answering, we do not need to compute it.

3. Motivation

Let us consider a VKG instance $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$ with the specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$. We are interested in ABox updates that consist of basic operations of two types, namely insertions and deletions of ABox facts. Formally, an ABox update is a pair $\mathcal{U}_A = \langle \mathcal{U}_A^+, \mathcal{U}_A^- \rangle$, where $\mathcal{U}_A^+$ denotes a finite set of *ABox insertions* and $\mathcal{U}_A^-$ denotes a finite set of *ABox deletions*, where each insertion and deletion is represented as an ABox assertion. We would like to understand how these ABox updates should be reflected in updates at the level of the data sources (DSs). Similarly to ABox updates, we also represent a data source update as a pair $U_D = \langle \mathcal{U}_D^+, \mathcal{U}_D^- \rangle$, where U_D^+ denotes a finite set of *data source insertions* and U_D^- denotes a finite set of *data source deletions*, each of them represented by a data source fact.

Let us consider for now the setting of a plain KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, for instance, a university's KB, which contains comprehensive information about students and their ongoing programs. Suppose that several students have changed their academic majors, and the update $\mathcal{U}_A$, is a list of these students with their new majors. This new data may conflict with some assertions in the ABox $\mathcal{A}$. For example, if in $\mathcal{T}$ no student can be registered in two different majors at the same time, and $\mathcal{A}$ asserts that *John* is a Computer Science (CS) major, but should be registered in Physics due to $\mathcal{U}_A$, then *John* can no longer be a CS major. To reconcile the conflict between the existing information in the KB and the new data in $\mathcal{U}_A$, we can update $\mathcal{A}$ to reflect that *John* actually has switched majors, which involves removing the outdated major in CS assignment. In most scenarios, to resolve inconsistencies that originate from changes at the instance level, it seems to be more intuitive to update the instance level rather than the intensional level. This means we would want the updated KB $\mathcal{K}'$ to be of the form $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where computing an ABox update results in a new ABox $\mathcal{A}'$, that along with the original TBox $\mathcal{T}$, represents the outcome of the update process.

However, in a VKG setting, a vital additional component is present – the *underlying data source*, $\mathcal{D}$, which is mapped to the ABox through a set $\mathcal{M}$ of declarative mappings. When ABox updates are triggered by real-world changes (like *John* changing his major), it is not only critical to maintain consistency in the ABox, but it is also crucial to ensure that these updates propagate to $\mathcal{D}$, preserving a harmonized state across the VKG system. In an ideal VKG framework featuring update propagation, when an update is performed at the ABox level, the changes should be automatically reflected in the underlying data source via the mappings. This notion implies a change to the structure of the VKG system from $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$ to $\mathcal{J}' = \langle \mathcal{P}, \mathcal{D}' \rangle$, highlighting the necessity of an update mechanism capable of propagating changes from ABox updates to the actual database. This kind of mechanism would considerably improve data consistency, reliability, and the practical utility of VKG systems in dynamic environments.

In this respect, we discuss in the next section three main goals to be achieved: First, we intend to formalize the framework of updates in the context of VKGs. Second, we delve into the task of computing updates at the ABox level of a VKG by propagating them to the underlying data source. And last, we propose an analysis of how schema mappings and ontology axioms affect updates in VKGs.

4. Updates in VKG systems

4.1. Updates in *DL-Lite*$_A$ KBs

The problem of instance-level updates in DL KBs has been studied extensively in the past [13, 14, 16, 15, 22], and since we are in a setting of incomplete information, insertion operations and their combination may generate an inconsistency in the ABox with respect to the axioms specified in the TBox[1].

Hence, the work on KB update is closely related to the work on inconsistency management under incomplete information. Three main strategies have been identified as ways to deal with inconsistency in such a context: *(i)* the first one consists of adjusting the constraints in the TBox $\mathcal{T}$ so that the updated ABox remains consistent; *(ii)* the second one, called *consistent query answering* [23], is concerned with answering queries while keeping the inconsistency in the KB;

[1]Note that, under the open world assumption typically holding in DL KBs, deletion operations instead do per se not generate inconsistencies.

(iii) the third one, which is more appropriate in our context (and in which we are interested), consists of removing facts from the original ABox to restore consistency [16, 17]. As shown in the literature [9, 24], whenever an update is applied over a KB, it is convenient to comply with the *minimal change principle*, which states that the resulting KB $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ should be as *close* as possible to the original one. In order to characterize the semantics of such a system, it becomes therefore necessary to rely on a suitably-defined consistency-tolerant semantics, e.g., based on the notion of repair.

In order to properly formalize the notion of KB updates, we need to define what it means for an update $\mathcal{U}_A$ to be *coherent* w.r.t. to a TBox $\mathcal{T}$. These notions are inspired by [14].

Definition 1 (Update coherence with $\mathcal{T}$). *Given a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, an update $\mathcal{U}_A = \langle \mathcal{U}_A^+, \mathcal{U}_A^- \rangle$ over $\mathcal{K}$ is* coherent *with the TBox $\mathcal{T}$ if (i) $Mod(\langle \mathcal{T}, \mathcal{U}_A^+ \rangle) \neq \emptyset$ and (ii) $cl_{\mathcal{T}}(\mathcal{U}_A^+) \cap \mathcal{U}_A^- = \emptyset$.*

The above definition exhibits the conditions required to be satisfied in order to validate the coherence of an update w.r.t the ontology. The first condition states that all the facts to be added should be consistent with the TBox $\mathcal{T}$ and the second condition is added to ensure that there is no insertion and deletion of the same fact. This definition leads us to the characterization of *accomplishment* of updates over a KB which is based on the notion of the *minimal change principle*

Definition 2 (Accomplishing an update). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB, $\mathcal{U}_A = \langle \mathcal{U}_A^+, \mathcal{U}_A^- \rangle$ an update coherent with $\mathcal{T}$, and $\mathcal{A}'$ an ABox. Then $\mathcal{A}'$* accomplishes *the update of $\mathcal{K}$ with $\mathcal{U}_A$ if (i) $\mathcal{A}'$ is $\mathcal{T}$-consistent, (ii) $\langle \mathcal{T}, \mathcal{A}' \rangle \not\models \beta$, for each $\beta \in \mathcal{U}_A^-$, and (iii) $\mathcal{A}' = \mathcal{A}'' \cup \mathcal{U}_A^+$ for some $\mathcal{A}'' \subseteq \mathcal{A}$.*

As we want to ensure that the resulting KB remains as *close* as possible to the original one, a semantic for the closeness measure has to be taken into account when comparing different resulting $\mathcal{A}'$ accomplishing $\mathcal{U}_A$ [25, 19, 26]. That's why the term *close* does not have a general definition that will fit with all circumstances when it comes to KB evolution. However, the notion of maximal accomplishment of an update has to be formalized, regardless of the chosen closeness measure.

Definition 3 (Maximally accomplishing an update). *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB, $\mathcal{U}_A = \langle \mathcal{U}_A^+, \mathcal{U}_A^- \rangle$ an update that is coherent with the TBox $\mathcal{T}$, and $\mathcal{A}'$ an ABox that accomplishes the update of $\mathcal{K}$ with $\mathcal{U}_A$. Then $\mathcal{A}'$* maximally accomplishes *the update of $\mathcal{K}$ with $\mathcal{U}_A$ if there is no $\mathcal{A}_1' \supsetneq \mathcal{A}'$ that accomplishes the update of $\mathcal{K}$ with $\mathcal{U}_A$.*

In the case of *DL-Lite*$_A$, it was shown in [26] that for any KB update $\mathcal{U}_A$ there is a *unique* ABox $\mathcal{A}'$ that maximally accomplishes it. For what follows, we denote by $\mathcal{K} \bullet \mathcal{U}_A$ the KB $\langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{A}'$ is the ABox that maximally accomplishes the update of $\mathcal{K}$ with $\mathcal{U}_A$.

The definitions introduced so far in this section are useful to formally define a KB update within the context of VKG systems $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$, where $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, as an update over $\langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle$. Computing such update essentially consists in computing the set of insertions and deletions of facts over $\mathcal{M}(\mathcal{D})$. In [14], a Datalog-based non-recursive algorithm has been proposed to derive the set of insertions (and deletions) of facts over $\mathcal{M}(\mathcal{D})$ from an update $\mathcal{U}_A$.

4.2. Updates over mapped data source

In order to maintain consistency between the knowledge graph and the data source, whenever a change is applied over the knowledge base we need to accurately propagate it down to the data source. This ensures that the source mirrors the requested change through the mappings. In this context, we have to extend the notion of KB updates to the VKG setting: we consider that the ABox facts in $\mathcal{A}$ are provided through the mapping $\mathcal{M}$ from a data source $\mathcal{D}$.

Since the knowledge graph is kept virtual, the only way to realize a KB update $\mathcal{U}_A$ is to find an appropriate database state $\mathcal{D}'$ from which it is possible to reach the updated ABox $\mathcal{A}'$. The translation of KB updates into suitable database (DB) updates is one of the key problems we are interested in this research, and to formalize it we first need to introduce the notion of *realizability* for KB updates.

Definition 4 (Realizability). *Let $\mathcal{J} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$ be a VKG instance forming a KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ with $\mathcal{A} = \mathcal{M}(\mathcal{D})$, and let $\mathcal{U}_A$ be a KB update coherent with $\mathcal{T}$. Then, $\mathcal{U}_A$ is* realizable *over $\mathcal{J}$ through $\mathcal{M}$ if there exists a DB instance $\mathcal{D}'$ such that $\mathcal{M}(\mathcal{D}') = \mathcal{K} \bullet \mathcal{U}_A$.*

In the context of VKGs, since mappings are seen as view functions over the underlying data source, the existence of a DB instance as specified in the above definition, is related to the notion of *translatability* of view updates described in [10].

However, due to the ambiguities provided by the presence mappings, there might exist multiple DB instances that realize a given KB update. Such ambiguity, which reflects the non-injectivity of the mappings, is defined in the following:

Definition 5 (Ambiguous Mapping). *A mapping $\mathcal{M}$ is said to be* ambiguous *if there exist two DB instances $\mathcal{D}_1$ and $\mathcal{D}_2$ s.t. (i) $\mathcal{D}_1 \neq \mathcal{D}_2$ and (ii) $\mathcal{M}(\mathcal{D}_1) = \mathcal{M}(\mathcal{D}_2)$.*

The above definition shows how the injectivity of the mapping is tightly connected to updates in VKG instances. Specifically, this tells us that there might exist multiple DB instances that accomplish a given KB update when the mapping component is not injective. However, whenever a mapping $\mathcal{M}$ is injective, every KB updates are uniquely translatable and is illustrated in the following proposition:

Definition 6 (Unique realizability). *Let $\mathcal{J} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$ be a VKG instance forming a KB $\mathcal{K} = \langle\mathcal{T}, \mathcal{A}\rangle$ with $\mathcal{A} = \mathcal{M}(\mathcal{D})$, and let $\mathcal{U}_A$ be a KB update coherent with $\mathcal{T}$. Then, $\mathcal{U}_A$ is* uniquely realizable *over $\mathcal{J}$ through $\mathcal{M}$ if there exists a unique DB instance $\mathcal{D}'$ such that $\mathcal{M}(\mathcal{D}') = \mathcal{K} \bullet \mathcal{U}_A$.*

Proposition 1. *Let $\mathcal{M}$ be a non-ambiguous mapping of VKG instance $\mathcal{J} = \langle\langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, \mathcal{D}\rangle$, and let $\mathcal{U}_A$ be an update coherent with $\mathcal{T}$. If $\mathcal{U}_A$ is realizable, then $\mathcal{U}_A$ is also uniquely realizable.*

Proof. It is clear that if two DB instances $\mathcal{D}_1$ and $\mathcal{D}_2$ realize a KB update $\mathcal{U}_A$, then we get $\mathcal{M}(\mathcal{D}_1) = \mathcal{M}(\mathcal{D}_2)$, which, by non-ambiguity of $\mathcal{M}$, entails $\mathcal{D}_1 = \mathcal{D}_2$. □

As stated in [27], the converse does not hold, i.e., even if $\mathcal{M}$ is ambiguous, a KB update $\mathcal{U}_A$ that is realizable might still be uniquely realizable. Therefore, in order to deal with the ambiguity provided by the presence of mappings in practical VKG systems (which represents a major concern in our research), we pose in the following sections the main research challenges we are facing, and any answer to them would benefit us in continuing our research.

4.2.1. Preference-based selection of DB instances

In a typical VKG instance, where the mapping component $\mathcal{M}$ is ambiguous, some KB updates will result in different DB instance candidates realizing them. As an example, if a mapping maps both the *Student* and *Professor* relations from the DB schema to the *Person* atom in the VKG, a KB update that adds a new person will result in two possible DB instances, one with an update to the *Student* relation and the other with an update to the *Professor* relation. To achieve automated propagation of updates to the data sources, one approach is to develop methods to automatically select the most suitable DB instance that realizes the desired VKG update based on some preferences initially provided by the users. We should note that all DB instances that realize a KB update are equivalent w.r.t. the mapping and the TBox, i.e., they create ABoxes that are equivalent with respect to the TBox. This notion of equivalence is defined as follows:

Definition 7 ($\mathcal{M}$-equivalence of DB instances)**.** *Given a VKG specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and two DB instances $\mathcal{D}_1$, $\mathcal{D}_2$ for $\mathcal{S}$, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are $\mathcal{M}$-equivalent w.r.t. $\mathcal{T}$ if $\mathcal{M}(\mathcal{D}_1) \equiv_{\mathcal{T}} \mathcal{M}(\mathcal{D}_2)$.*

In order to reduce the space of possible DB instances that realize a given KB update, we take advantage of the notion of *repair* by complying with the *minimal change principle* w.r.t. the original instance. Moreover, we want to base our selection of repair on a set of preferences provided by the user, as explained in [28]. This brings us to the following research questions:

Q1: *How can we formalize a preference-based selection of DB instances that realize updates over VKGs?*

Q2: *Given a VKG instance $\mathcal{J} = \langle \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle, \mathcal{D} \rangle$, how can we formalize a suitable notion of* interference *of predicates over $\mathcal{D}$ that may occur when propagating any given KB update, and how can we actually compute such interferences?*

For instance, in our previous example, the relations *Student* and *Professor* are interfering since they map to the same atom in the KB.

The above research question is helpful to check whether a priority operator based on preferences given by the user is *total*, in the sense that it resolves all interferences. This is because a preference that cannot be further extended should specify how to resolve every interference [28]. This leads to a further research question:

Q3: *What language is suitable to express user preferences in the context of updates in VKGs?*

Finally, we also want to consider the case where KB updates are not realizable, i.e., given a KB $\mathcal{K}$ updated with $\mathcal{U}_A$, there is no DB instance $\mathcal{D}$ such that $\mathcal{M}(\mathcal{D}) = \mathcal{K} \bullet \mathcal{U}_A$. The absence of realizability entails the presence of DB instances that are not $\mathcal{M}$-equivalent and that create different approximations of $\mathcal{K} \bullet \mathcal{U}_A$. This adds a further challenge, which consists of choosing the DB instance $\mathcal{D}'$ such that the KB $\mathcal{K}' = \langle \mathcal{T}, \mathcal{M}(\mathcal{D}') \rangle$ is *minimally distant* from $\mathcal{K} \bullet \mathcal{U}_A$, for a suitably defined notion of distance. Therefore, we need to provide a *relaxed* version of realizability, which consists of finding a DB instance that *realizes at best* a given KB update $\mathcal{U}_A$. This brings us to the following research question:

Q4: *Given a non-realizable KB update $\mathcal{U}_A$ over a KB $\mathcal{K}$, how can we select the DB instance that through $\mathcal{M}$ approximates at best $\mathcal{K} \bullet \mathcal{U}_A$?*

4.2.2. KB update rewriting in VKGs

In this section, we are interested in statically analyzing updates in the context of VKGs, i.e., given a VKG specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a KB update $\mathcal{U}_A$, we want to compute a suitable DB update $\mathcal{U}_D$ that is its translation and that updates any data source in such a way that via the mapping it generates the updated KB. Following [10], which deals with view-updates, such a translation can be formalized as follows.

Definition 8 (Translation). *Given a VKG specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a KB update $\mathcal{U}_A$, a DB update $\mathcal{U}_D$ is a* translation of $\mathcal{U}_A$ *if for every DB instance $\mathcal{D}$ satisfying $\mathcal{S}$ from which we have $\mathcal{K} = \langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle$, we have that:* (1) $\mathcal{K} \bullet \mathcal{U}_A = \langle \mathcal{T}, \mathcal{M}(\mathcal{D} \bullet \mathcal{U}_D) \rangle$ *and* (2) $\mathcal{D} \bullet \mathcal{U}_D = \mathcal{D}$ *whenever* $\mathcal{K} \bullet \mathcal{U}_A = \mathcal{K}$.

Based on the above definition, we want to be able to statically compute the translation (or set of translations) of a given KB update, which is composed of a set of insertions and deletions of relations over the data source.

Unlike the method proposed in [14], which is based on a Datalog Program that considers a DB instance, we want to propose a static method that supports the query rewriting algorithm PerfectRef [7] in order to entail the set of insertions/deletions over the KB. Finally, we want to unfold each KB insertion/deletion w.r.t. to a given mapping in order to compute the corresponding update over the data source.

In this setting, we want to formalize the notion of *static realizability*, which is defined as follows:

Definition 9 (Static realizability). *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification and $\mathcal{U}_A$ a KB update. Then, we say that $\mathcal{U}_A$ is* statically realizable *if there exists a DB update $\mathcal{U}_D$ over the signature $\mathcal{S}$ such that $\langle \mathcal{T}, \mathcal{M}(\mathcal{D} \bullet \mathcal{U}_D) \rangle = \langle \mathcal{T}, \mathcal{M}(\mathcal{D}) \rangle \bullet \mathcal{U}_A$, for every DB instance $\mathcal{D}$ compliant with $\mathcal{S}$.*

Given this notion of static realizability, we can now formulate a further research question:

Q5: *How can KB updates be statically and effectively realized in state-of-the-art VKGs that support query rewriting?*

4.2.3. Mapping analysis in VKGs

Finally, we want to study mapping analysis in the scope of VKGs, especially its inherent properties that are intrinsically connected to VKG updates. The task of mapping specification within the context of VKGs can be complex, necessitating a deep understanding of both the ontology and the associated data source. This nuanced "cognitive distance" between the data sources and the ontology often prompts the formulation of a complex mapping, expressed via assertions that correlate queries over the ontology with those over the data sources. A particular interest to us is the concept of *injectivity* of schema mapping—a facet that bears significant implications for the translation of KB updates into DB updates within the framework of VKG.

The *injectivity* property of the mapping is particularly crucial for updates, as it ensures that each change in the ontology corresponds to a unique change in the DB. If a mapping is not injective, a single update at the ontology level could potentially correspond to multiple updates at the DB level, creating ambiguity in the translation process. This ambiguity could lead to inconsistent states between the ontology and the DB, undermining the overall integrity of the system. However, the absence of injectivity in the schema mapping does not entirely prevent the translation of updates. There may be situations where updates can be uniquely translated even if the schema mapping is not injective. Still, such scenarios are more challenging to manage and typically require additional mechanisms or constraints to ensure the correct translation of updates.

As a first step towards analyzing mapping in VKG, we want to determine whether a given mapping specification is injective (or ambiguous) and in order to achieve it we are planning to use an automated model prover.

Q6: *Given a VKG specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, how can we check whether the mapping component $\mathcal{M}$ is injective?*

To address this research question, our planned approach is to harness the capabilities of automated tools, such as the Z3 model prover-a highly proficient theorem prover developed under the auspices of Microsoft Research [29]. The ultimate aim is to integrate this tool into the Ontop system infrastructure for mapping analysis.

Also, we express interest in non-injective mappings, particularly in deriving the necessary information from the system to facilitate a deterministic translation of KB updates. This is tightly connected to work on unique solutions in data exchange, as studied in [30] where we want to ensure that for any subjective mapping, there is always a unique data source state from which it's possible to compute a given KB. This is formulated in the following research question:

Q7: *Given a VKG specification $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ where $\mathcal{M}$ is not injective, what additional information is necessary in order to maintain consistency between the knowledge base and the DB via the mapping?*

5. Conclusions

This paper aims to address foundational and applied issues in the realm of Virtual Knowledge Graphs (VKGs), focusing on the introduction and exploration of Ontology-based updates and evolution within the context. We explored the multifaceted issue of translating knowledge base updates into database updates in the setting of VKGs. The crux of our research challenges stems from the dual nature of the problem: maintaining consistency in an incomplete information context and the inherent ambiguity introduced by mappings between ontology and data sources.

Regarding the research questions mentioned in this paper, we have already obtained some preliminary results. However, these methods require extensive discussion and need to be further analyzed to ascertain their validity and the potential for being extended.

Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and by the Province of Bolzano and DFG through project D2G2 (DFG grant no. 500249124). Šimkus was partially supported by the Austrian Science Fund (FWF) project P30873.

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. on Data Semantics 10 (2008) 133–173. doi:10.1007/978-3-540-77688-8_5.

[2] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: The *DL-Lite* approach, in: S. Tessaris, E. Franconi (Eds.), Reasoning Web: Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW), volume 5689 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 255–356.

[3] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI), IJCAI Org., 2018, pp. 5511–5519. doi:10.24963/ijcai.2018/777.

[4] G. Xiao, L. Ding, B. Cogrel, D. Calvanese, Virtual Knowledge Graphs: An overview of systems and use cases, Data Intelligence 1 (2019) 201–223. doi:10.1162/dint_a_00011.

[5] M. Krötzsch, OWL 2 profiles: An introduction to lightweight ontology languages, in: Reasoning Web: Semantic Technologies for Advanced Query Answering – 8th Int. Summer School Tutorial Lectures (RW), volume 7487 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 112–183. doi:10.1007/978-3-642-33158-9_4.

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

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family, J. of Automated Reasoning 39 (2007) 385–429. doi:10.1007/s10817-007-9078-x.

[8] H. Katsuno, A. Mendelzon, On the difference between updating a knowledge base and revising it, in: Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR), 1991, pp. 387–394.

[9] T. Eiter, G. Gottlob, On the complexity of propositional knowledge base revision, updates and counterfactuals, Artificial Intelligence 57 (1992) 227–270.

[10] F. Bancilhon, N. Spyratos, Update semantics of relational views, ACM Trans. on Database Systems 6 (1981) 557–575.

[11] S. S. Cosmadakis, C. H. Papadimitriou, Updates of relational views, J. of the ACM 31 (1984) 742–760. doi:10.1145/1634.1887.

[12] A. C. Kakas, P. Mancarella, Database updates through abduction, in: Proc. of the 16th Int. Conf. on Very Large Data Bases (VLDB), volume 90, 1990, pp. 650–661.

[13] D. Calvanese, E. Kharlamov, W. Nutt, D. Zheleznyakov, Updating ABoxes in *DL-Lite*, in: Proc. of the 4th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW), volume 619 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2010, pp. 3.1–3.12.

[14] G. De Giacomo, X. Oriol, R. Rosati, D. F. Savo, Instance-level update in DL-Lite ontologies through first-order rewriting, J. of Artificial Intelligence Research 70 (2021) 1335–1371. doi:10.1613/jair.1.12414.

[15] H. Liu, C. Lutz, M. Milicic, F. Wolter, Updating description logic ABoxes, in: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR), 2006, pp.

46–56.

[16] G. De Giacomo, M. Lenzerini, A. Poggi, R. Rosati, On instance-level update and erasure in description logic ontologies, J. of Logic and Computation 19 (2009) 745–770.

[17] M. Winslett, A model-based approach to updating databases with incomplete information, ACM Trans. on Database Systems 13 (1988) 167–196.

[18] M. Lenzerini, D. F. Savo, Updating inconsistent description logic knowledge bases, in: Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI), 2012, pp. 516–521.

[19] M. L. Ginsberg, D. E. Smith, Reasoning about action I: A possible worlds approach, Artificial Intelligence 35 (1988) 165–195.

[20] R. Fagin, J. D. Ullman, M. Y. Vardi, On the semantics of updates in databases, in: Proc. of the 2nd ACM Symp. on Principles of Database Systems (PODS), 1983, pp. 352–365.

[21] M. Lenzerini, Data integration: A theoretical perspective., in: Proc. of the 21st ACM Symp. on Principles of Database Systems (PODS), 2002, pp. 233–246. doi:10.1145/543613.543644.

[22] D. Zheleznyakov, E. Kharlamov, W. Nutt, D. Calvanese, On expansion and contraction of DL-Lite knowledge bases, J. of Web Semantics 57 (2019) 1–19. doi:10.1016/j.websem.2018.12.002.

[23] L. Bertossi, Database Repairs and Consistent Query Answering, Morgan & Claypool Publishers, 2011.

[24] G. Flouris, D. Plexousakis, Handling Ontology Change: Survey and Proposal for a Future Research Direction, Technical Report TR-362 FORTH-ICS, Institute of Computer Science, Forth, Greece, 2005.

[25] M. L. Ginsberg, Counterfactuals, Artificial Intelligence 30 (1986) 35–79.

[26] D. Calvanese, E. Kharlamov, W. Nutt, D. Zheleznyakov, Evolution of *DL-Lite* knowledge bases, in: Proc. of the 9th Int. Semantic Web Conf. (ISWC), volume 6496 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 112–128. doi:10.1007/978-3-642-17746-0_8.

[27] E. Franconi, P. Guagliardo, On the translatability of view updates, in: Proc. of the 6th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW), volume 866 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2012, pp. 154–167.

[28] S. Staworko, J. Chomicki, J. Marcinkowski, Prioritized repairing and consistent query answering in relational databases, Ann. of Mathematics and Artificial Intelligence 64 (2012) 209–246. doi:10.1007/s10472-012-9288-8.

[29] L. De Moura, N. Bjørner, Z3: An efficient SMT solver, in: Proc. of the Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2008, pp. 337–340.

[30] N. Ngo, E. Franconi, Unique solutions in data exchange, in: Proc. of the 25th Int. Conf. on Database and Expert Systems Applications (DEXA), volume 8645 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 281–294.