Ontology-Based Update in Virtual Knowledge Graphs via Schema Mapping Recovery

Romuald Esdras Wandji[1(✉)] and Diego Calvanese[1,2]

[1] Department of Computing Science, Umeå Universitet, Umeå, Sweden
{romuald.esdras.wandji,diego.calvanese}@umu.se, diego.calvanese@unibz.it
[2] Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy

Abstract. In Virtual Knowledge Graphs (VKGs), access to a relational data source is provided through an ontology, which is linked to the data source via declarative mappings. VKGs stand as a predominant paradigm for the access to (and integration of) heterogeneous data sources. However, little attention has been paid so far to the issue of updates in VKGs expressed over the ontology, which represents a crucial feature for fully managing data sources through the lens of an ontology. In this paper, we consider the problem of updating a VKG instance by specifying a set of insertions and deletions of ontology instances and propagating these updates to the underlying data source through the VKG mapping. We consider ontologies specified in the *DL-Lite*$_R$ lightweight ontology language and study the problem for the case where source queries in mappings are unions of conjunctive queries. We rely on the notion of maximum recovery of VKG mappings, borrowed from the data exchange setting, and propose methods to compute the set of source updates that translate an ontology update with a minimal side-effect, considering both insertions and deletions of multiple ABox assertions.

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

1 Introduction

Virtual Knowledge Graphs (VKGs) provide a powerful paradigm to address data integration, which has been studied extensively for the case of relational data sources. In VKGs, users interact with a high-level, conceptual representation of the domain of interest given in the form of an ontology that is linked to the data source via declarative mappings [6,23,25,26]. The ontology is typically expressed in the OWL 2 QL profile of the Web Ontology Language (OWL 2) [22]. Such language, which has been specifically designed to be used in this context, has its formal counterpart in *DL-Lite*$_R$, a Description Logic (DL) of the *DL-Lite* family of lightweight DLs [7].

The primary focus of research on VKGs has been on query answering i.e., the problem of computing the answers to a user query posed over the ontology by exploiting the ontology axioms and the mapping layer to extract the relevant data from the underlying data source. In this paper, instead, we are concerned with the challenging problem of *updating* a VKG by operating on the ontology, which is a key task required in full-fledged ontology-based data management systems, where all operations that involve the

S. Kirrane et al. (Eds.): RuleML+RR 2024, LNCS 15183, pp. 59–74, 2024.
https://doi.org/10.1007/978-3-031-72407-7_6

data source should be carried out through the lens of the ontology. Notice that this problem requires on the one hand to deal with potential inconsistencies between the update and the axioms of the ontology TBox, and on the other hand to propagate the update via the mappings to the underlying data source. The first problem is rooted in the vast literature on knowledge base update and belief revision [15,19,20,24], which includes work dealing specifically with lightweight DLs [27]. The latter problem, instead, is tightly connected to the problem of *view-update*, which has a long history in the database literature [4,12,18], but is still open in the general case. Instead, only very few works have dealt with how mappings impact updates in the context of VKGs. In [14], it is proposed to store ontology updates in additional tables, without affecting the original database, and by adjusting the mapping to reconstruct the updated VKG from the original one and the data in the additional tables. However, such approach cannot be applied in the very common case where the user has access to the data sources at the data level, but is not entitled to modify their schema or to add additional sources.

In this work, we are therefore interested in how to perform updates in VKGs by directly propagating them to the underlying data source via the mappings. For this, we have to address two key problems: *(i)* The update might not be *realizable*, since an insertion on the data source guaranteeing that the requested ABox facts are inserted from the VKG, might have as *side-effect* the insertion of additional ABox facts. Similarly for deletions. *(ii)* The mappings might lose information (e.g., due to the presence of disjunction or of existential variables in mapping queries), and therefore there might be multiple ways to update the source data corresponding to a VKG update.

As a motivating example, consider a data source with two relations: $RS(res, sup)$ that relates researchers to their supervisor(s), and $SG(sup, gr)$ that relates supervisors to the grants they have access to. The information in this source has to be integrated into a VKG whose ontology contains a role $access$, relating researchers to grants, and a class $Active$ containing active supervisors. We consider a mapping $\mathcal{M}$ between this source schema and the ontology consisting of the following assertion:

$$\exists y.RS(x, y) \wedge SG(y, z) \rightsquigarrow access(x, z). \tag{1}$$

Let us assume a user wants to update the extensional content of the system by inserting a new grant access with the ABox fact $access(\texttt{john}, \texttt{grant1})$. Due to the existential variable in the mapping source query, there are (infinitely) many distinct translations of the given ABox insertion, e.g., the insertions in the data source of $\{RS(\texttt{john}, \texttt{sup1}), SG(\texttt{sup1}, \texttt{grant1})\}$, or of $\{RS(\texttt{john}, \texttt{sup2}), SG(\texttt{sup2}, \texttt{grant1})\}$. But in reality, which supervisor to choose for the researcher being inserted has no importance in this context as this information is not visible to the user. So, it can be randomly created, or a default value can be assigned to every added access. However, if $SG(x, y) \rightsquigarrow Active(x)$ is added to $\mathcal{M}$, the choice of the supervisor for an inserted access becomes important, because it may lead to an unintended effect over the VKG. Depending on other facts in the source, inserting a new grant access might lead to a side-effect, thus making the update unrealizable.

In this paper, we address this problem by proposing a comprehensive framework for ABox updates in VKGs that builds on the work on *schema mapping recovery* introduced in the context of data exchange [2] (see Sect. 4), and on *data lineage* in databases [9,10]

(see Sect. 5). Specifically, we show that we can use a so-called *maximum recovery* to characterize the source updates corresponding to ABox updates, and moreover, such maximal recoveries can be computed from the mapping assertions using query rewriting techniques. Exploiting these notions, we provide algorithms for translating a set of ABox deletions and insertions into suitable deletions or insertions over the source with minimal side effects (see Sect. 6). Section 7 concludes the paper.

2 Preliminaries

We now introduce the notions on DLs, databases (DBs), and VKGs necessary to understand the technical development in the paper, assuming familiarity with the syntax and semantics of first-order logic (FOL). We consider distinct, countably infinite, and pairwise disjoint alphabets $\mathbf{N}_C$ of *concept names*, $\mathbf{N}_P$ of *role names*, and $\mathbf{N}_I$ of constants.

2.1 Description Logic Knowledge Bases

Description Logics (DLs) [3] allow for modeling a domain of interest in terms of *concepts* and *roles*, which correspond to unary and binary predicates, respectively. A DL *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a *TBox* $\mathcal{T}$, capturing intensional information, and an *ABox* $\mathcal{A}$, providing extensional information. We consider DLs of the *DL-Lite* family [7,23], and specifically *DL-Lite*$_R$, which is the formal counterpart of the tractable OWL 2 QL profile of the Web Ontology Language (OWL 2) [22].

A *DL-Lite*$_R$ TBox is a finite set of 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*), or $R_1 \sqsubseteq \neg R_2$ (*role disjointness*). Here, R (possibly sub-scripted) denotes an atomic role $P \in \mathbf{N}_P$ or its inverse P^-, while B (possibly sub-scripted) denotes a *basic concept*, which is either an atomic concept $A \in \mathbf{N}_C$, or a concept of the form $\exists R$. For a TBox $\mathcal{T}$, we use $\mathcal{T}^+$ to denote the set of concept and role inclusions in $\mathcal{T}$, and $\mathcal{T}^-$ to denote the set of concept and role disjointness assertions in $\mathcal{T}$, hence $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$. A *DL-Lite*$_R$ ABox is a finite set of assertions of the form $A(c)$ or $P(c, c')$, with $A \in \mathbf{N}_C$, $P \in \mathbf{N}_P$, and $c, c' \in \mathbf{N}_I$.

The semantics of a DL KB is given in terms of first-order interpretations [3], where 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 $A \in \mathbf{N}_C$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $P \in \mathbf{N}_P$ to $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each $c \in \mathbf{N}_I$ to $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation functions is extended to arbitrary concepts and roles as follows: $(P^-)^{\mathcal{I}} = \{(o', o) \mid (o, o') \in P^{\mathcal{I}}\}$, $(\exists R)^{\mathcal{I}} = \{o \mid \text{there exists } o' \text{ s.t. } (o, o') \in R^{\mathcal{I}}\}$, $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$, and $(\neg R)^{\mathcal{I}} = (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of (or *satisfies*) $A(c)$ if $c^{\mathcal{I}} \in A^{\mathcal{I}}$, $P(c, c')$ if $(c^{\mathcal{I}}, c'^{\mathcal{I}}) \in P^{\mathcal{I}}$), and $E_1 \sqsubseteq E_2$ if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$. Also, $\mathcal{I}$ is a *model* of $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of all assertions in $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{T} \cup \mathcal{A}$, respectively. We denote with $Mod(\mathcal{K})$ the set of all models of $\mathcal{K}$, and say that $\mathcal{K}$ is *consistent* if $Mod(\mathcal{K}) \neq \emptyset$. We say that $\mathcal{A}$ is $\mathcal{T}$*-consistent*, if $\langle \mathcal{T}, \mathcal{A} \rangle$ is consistent. A TBox or ABox assertion α is *logically implied* by $\mathcal{T}$ (resp., $\mathcal{K}$), denotes $\mathcal{T} \models \alpha$ (resp., $\mathcal{K} \models \alpha$) if each model of $\mathcal{T}$ (resp., $\mathcal{K}$) satisfies α. 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 assertions that

are logically implied by $\mathcal{T}$. Finally, given two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$, we say that $\mathcal{A}_1$ is *logically equivalent* to $\mathcal{A}_2$ w.r.t. $\mathcal{T}$, denoted $\mathcal{A}_1 \equiv_{\mathcal{T}} \mathcal{A}_2$, if $cl_{\mathcal{T}}(\mathcal{A}_1) = cl_{\mathcal{T}}(\mathcal{A}_2)$.

We adopt here the *standard name assumption*, i.e., for each interpretation $\mathcal{I}$ of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\Delta^{\mathcal{I}}$ contains all individuals of $\mathcal{A}$, and for each such individual c, $c^{\mathcal{I}} = c$.

2.2 Relational Databases and Queries

A *database schema* is a finite set $\mathcal{S} = \{r_1/n_1, \ldots, r_k/n_k\}$ of relation schemas, where each r_i is a predicate name of arity n_i. A database instance D over $\mathcal{S}$ maps each predicate r/n in $\mathcal{S}$ to an n-ary relation, denoted r^D. An *atom* for r/n has the form $r(t_1, \ldots, t_n)$, or simply $r(\boldsymbol{t})$, where each t_j is a term, which can be a constant from $\mathbf{N}_I$ or a variable. If all t_j's are constants, the atom is called ground, or simply a *tuple*.

A (FOL) formula over a relational schema $\mathcal{S}$ is constructed by using the relation names in $\mathcal{S}$, the equality predicate $=$, and the constants in $\mathbf{N}_I$. A formula φ with free variables $\boldsymbol{x}$ is denoted $\varphi(\boldsymbol{x})$, and is also called a (relational) *query* with *answer variables* $\boldsymbol{x}$. The formula is *closed* when $\boldsymbol{x}$ is empty. The closed formula obtained from $\varphi(\boldsymbol{x})$ by replacing every variable in $\boldsymbol{x}$ by the corresponding constant in $\boldsymbol{c}$ is denoted $\varphi(\boldsymbol{c})$. For an instance D over $\mathcal{S}$, we use $D \models \varphi(\boldsymbol{c})$ to denote that $\varphi(\boldsymbol{c})$ holds in D. For a closed formula φ and a set Σ of closed formulae over $\mathcal{S}$, we use $\Sigma \models \varphi$ to indicate that φ *is implied by* Σ, i.e., φ holds in all instances in which all formulae of Σ hold.

A *conjunctive query* (CQ) $Q(\boldsymbol{x})$ over the schema $\mathcal{S}$ is a query defined by a formula of the form $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$, where $\varphi = r_1(\boldsymbol{t}_1) \wedge \cdots \wedge r_n(\boldsymbol{t}_n)$ is a conjunction of atomic formulae whose variables belong to $\boldsymbol{x} \cup \boldsymbol{y}$. The variables $\boldsymbol{y}$ are called *existential variables*. We also express such CQ as a logical rule of the form $Q(\boldsymbol{x}) \leftarrow r_1(\boldsymbol{t}_1), \ldots, r_n(\boldsymbol{t}_n)$, where $Q(\boldsymbol{x})$ is called the *head* and $r_1(\boldsymbol{t}_1), \ldots, r_n(\boldsymbol{t}_n)$ the *body* of the rule. A *union of conjunctive queries* (UCQ) is a finite disjunction of CQs with the same answer variables, also represented as a finite set of rules with the same head.

2.3 Schema Mappings

In VKGs, mappings are used to (virtually) populate ontology concepts and roles with individuals and values derived from the data in the underlying data source. We consider mappings specified in the standard language R2RML [11], but adopt a simplified abstract notation [23], which is also close to the one of *plain second-order tuple-generating dependency* (plain SO-tgds) introduced in data exchange [1].

Formally, a *VKG mapping* $\mathcal{M}$ from a source schema $\mathcal{S}$ to an ontology (TBox) $\mathcal{T}$ consists of a set of *mapping assertions* of the form $Q(\boldsymbol{x}) \rightsquigarrow E(\boldsymbol{t}(\boldsymbol{x}))$, where $Q(\boldsymbol{x})$ is a CQ (called *source query*) over $\mathcal{S}$ of arity $n > 0$, and $E(\boldsymbol{t}(\boldsymbol{x}))$ is an atom (called *target atom*) over $\mathcal{T}$ with variables in $\boldsymbol{x}$. Such atom has the form $A(t_1(\boldsymbol{x}_1))$, with $A \in \mathbf{N}_C$, or $P(t_1(\boldsymbol{x}_1), t_2(\boldsymbol{x}_2))$, with $P \in \mathbf{N}_P$, where the terms $t_1(\boldsymbol{x}_1)$ and $t_2(\boldsymbol{x}_2)$ denote so-called *IRI-templates*, obtained by applying (skolem) functions to the answer variables of the source query. Such IRI-templates[1] are used to generate strings representing object *IRIs* (Internationalized Resource Identifiers) or (RDF) literals, starting

[1] IRI-templates correspond to the R2RML *string templates*.

from DB values retrieved by the source query in the mapping. Notice that in practice, it is desired that the union of all templates corresponds to an *injective* function, i.e., there is a unique way to reconstruct from a string s, the actual IRI-template $t(\boldsymbol{x})$ and the constituent values $\boldsymbol{v}$ used to generate $s = t(\boldsymbol{v})$. Moreover we assume that for each IRI-template $f(x_1, \ldots, x_n)$, we have also n *inverse templates*[2] $f^1, \ldots, f^n$ such that $f^i(f(v_1, \ldots, v_n)) = v_i$, for each $i \in [1..n]$ and for all possible values $v_1, \ldots, v_n$ instantiating the variables in $t(x_1, \ldots, x_n)$.

Such a VKG mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$, which we also call a *source-to-ontology mapping* (*so-mapping*), maps a DB instance D of $\mathcal{S}$ to the (*unique*) ABox

$$\mathcal{M}(D) = \{\mathsf{as}(E(\boldsymbol{t}(\boldsymbol{x})), \boldsymbol{x} \mapsto \boldsymbol{o}) \mid (\boldsymbol{x} \mapsto \boldsymbol{o}) \in Q(\boldsymbol{x})^D, (Q(\boldsymbol{x}) \rightsquigarrow E(\boldsymbol{t}(\boldsymbol{x}))) \in \mathcal{M}\}$$

where $\boldsymbol{x} \mapsto \boldsymbol{o}$ represents a solution mapping from the evaluation $Q(\boldsymbol{x})^D$ of the source query $Q(\boldsymbol{x})$ over the DB instance D, and the term $\mathsf{as}(E(\boldsymbol{t}(\boldsymbol{x})), \boldsymbol{x} \mapsto \boldsymbol{o}))$ denotes the ABox assertion obtained by applying the solution mapping $\boldsymbol{x} \mapsto \boldsymbol{o}$ to $E(\boldsymbol{t}(\boldsymbol{x}))$. This application typically involves replacing $\boldsymbol{x}$ with $\boldsymbol{o}$ in the templates in $\boldsymbol{t}(\boldsymbol{x})$ (using appropriate string concatenation operations). It is also convenient to consider an so-mapping $\mathcal{M}$ as the set of pairs $\{(D, \mathcal{M}(D)) \mid D \text{ is a DB instance of } \mathcal{S}\}$, so that $(D, \mathcal{A}) \in \mathcal{M}$ is an alternative notation for $\mathcal{A} = \mathcal{M}(D)$. Notice that, since a concept name A (or role name P) might appear as target atom of multiple mapping assertions, the source query that generates the instances of A (or P) is in general a UCQ.

2.4 Virtual Knowledge Graphs

We recall the main elements of the VKG framework, also known as *Ontology-based Data Access* (OBDA) [23,25]. A *VKG specification* is a tuple $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, where $\mathcal{T}$ is a *DL-Lite*$_R$ TBox (typically expressed in OWL 2 QL), $\mathcal{S}$ is a data source schema, and $\mathcal{M}$ is a set of so-mappings from $\mathcal{S}$ to $\mathcal{T}$. A *VKG instance* is a pair $\langle \mathcal{P}, D \rangle$ where D is a source instance of $\mathcal{S}$. Its semantics is defined in terms of FO interpretations over $\mathcal{T}$. An interpretation $\mathcal{I}$ is a *model* of $\langle \mathcal{P}, D \rangle$ if it is a model of the DL KB $\langle \mathcal{T}, \mathcal{M}(D) \rangle$.

We recall also the notion of *mapping saturation* [21], which allows one to compile into the mapping $\mathcal{M}$ the inclusion assertions of $\mathcal{T}$ that do *not* involve an existential quantification $\exists R$ in the right-hand side. Specifically, to compute the *mapping saturation* $sat_{\mathcal{T}}(\mathcal{M})$ *of* $\mathcal{M}$ *w.r.t.* $\mathcal{T}$, we start from $\mathcal{M}$, and repeatedly add implied mapping assertions. E.g., if $Q(\boldsymbol{x}) \rightsquigarrow A_1(t(\boldsymbol{x}))$ is in $sat_{\mathcal{T}}(\mathcal{M})$ and $A_1 \sqsubseteq A_2$ is in $\mathcal{T}$, we add to $sat_{\mathcal{T}}(\mathcal{M})$ also $Q(\boldsymbol{x}) \rightsquigarrow A_2(t(\boldsymbol{x}))$; similarly for inclusions $\exists R \sqsubseteq A$ and $R_1 \sqsubseteq R_2$.

3 Instance-Level Updates in VKGs

We consider instance-level updates that consist of basic operations of two types, namely deletions and insertions of ABox assertions, and the key problem we are interested in is the translation of ABox updates into suitable source updates. In this paper, we restrict our attention to the two cases of updates consisting either of a set $\mathcal{U}_A^-$ of *ABox deletions*, or of a set $\mathcal{U}_A^+$ of *ABox insertions*, each given as a set of ABox facts, and consider

[2] These correspond to the `rr:inverseExpression` of R2RML.

translations that are of the same type as the ABox updates, i.e., $\mathcal{U}_A^-$ translates to a set $\mathcal{U}_D^-$ of source deletions, and $\mathcal{U}_A^+$ to a set $\mathcal{U}_D^+$ of source insertions. We leave the case of combined deletions and insertions and of more general translations for future work.

We concentrate here on the problem of propagating updates to the data source via the mappings and do not deal explicitly with possible inconsistencies w.r.t. the TBox and (virtual) ABox that an update might cause. Hence, we take the simple approach of rejecting updates that would lead to an inconsistency. Since for *DL-Lite*$_R$, inconsistencies are due to the violation of disjointness assertions only and deletions of ABox facts cannot cause such violations, we need to pay attention to inconsistency only for ABox insertions. Formally, we reject an insertion update $\mathcal{U}_A^+$ if $\langle \mathcal{T}, \mathcal{A} \cup \mathcal{U}_A^+ \rangle$ is inconsistent. Instead of rejecting inconsistent updates, we could also rely on the many approaches that have been proposed for handling inconsistency in KB updates, which typically are based on computing some form of *ABox repair* that removes the inconsistencies from the KB [19,20,24]. We can then consider the original update combined with the ABox repair as the overall consistent VKG update to which we apply our techniques.

Apart from possible inconsistencies, the TBox has additional effects on updates that we need to take into account. Since *DL-Lite*$_R$ is Horn [7], an entailed ABox fact is already entailed by the TBox $\mathcal{T}$ together with a *single ABox fact* [27]. For an ABox insertion $\mathcal{U}_A^+$, we can, therefore account for $\mathcal{T}$ by adding to $\mathcal{U}_A^+$ all facts in $cl_{\mathcal{T}}(\mathcal{U}_A^+)$. For an ABox deletion $\mathcal{U}_A^-$, instead, we can account for $\mathcal{T}$ by (repeatedly) adding to $\mathcal{U}_A^-$ all those facts from the ABox $\mathcal{A}$ that would entail a fact already in $\mathcal{U}_A^-$. We denote such set of facts by $invcl_{\mathcal{T}}^{\mathcal{A}}(\mathcal{U}_A^-)$. Notice that $cl_{\mathcal{T}}(\mathcal{U}_A^+)$ and $invcl_{\mathcal{T}}^{\mathcal{A}}(\mathcal{U}_A^-)$ can be computed in polynomial time in the combined size of $\mathcal{U}_A^+$, $\mathcal{U}_A^-$, $\mathcal{T}$, and $\mathcal{A}$ [7,27]. Based on this, we will assume in the following that $\mathcal{U}_A^+$ is already closed w.r.t. $\mathcal{T}$, i.e., that $\mathcal{U}_A^+ = cl_{\mathcal{T}}(\mathcal{U}_A^+)$, and that $\mathcal{U}_A^-$ is already "inverse-closed" w.r.t. $\mathcal{T}$ and $\mathcal{A}$, i.e., that $\mathcal{U}_A^- = invcl_{\mathcal{T}}^{\mathcal{A}}(\mathcal{U}_A^-)$.

Considering (inverse-)closed updates brings us nearer to the case where we can restrict the attention to VKG specifications where the TBox does not contain any inclusion assertion, but only disjointness assertions. However, we have to consider that propagating ABox updates to the data source might lead to source updates that have side-effects in the ABox. Since we are interested in minimizing such side-effects, we also have to take into account how they are affected by the TBox. In the case of insertions, the TBox axioms might amplify side-effects, while for deletions, they might amplify them but also reduce them, as illustrated in the following example.

Example 1. Let $\mathcal{P}_0 = \langle \mathcal{T}_0, \mathcal{M}_0, \mathcal{S}_0 \rangle$ be a VKG specification, where $\mathcal{T}_0 = \{A_1 \sqsubseteq B_1, A_1 \sqsubseteq B_2\}$, $\mathcal{S}_0 = \{r_1/1, r_2/1, r_3/1\}$, and $\mathcal{M}_0 = \{r_1(x) \rightsquigarrow A_1(x), r_2(x) \rightsquigarrow B_1(x), r_3(x) \rightsquigarrow A_2(x)\}$, and let $\mathcal{J}_0 = \langle \mathcal{P}_0, D_0 \rangle$ be a corresponding VKG instance, where $D_0 = \{r_1(\mathsf{c}), r_2(\mathsf{c}), r_3(\mathsf{c})\}$. We obtain $\mathcal{M}_0(D_0) = \{A_1(\mathsf{c}), B_1(\mathsf{c}), A_2(\mathsf{c})\}$, and $cl_{\mathcal{T}_0}(\mathcal{M}_0(D_0)) = \mathcal{M}_0(D_0) \cup \{B_2(\mathsf{c})\}$. Suppose we want to delete the ABox fact $B_1(\mathsf{c})$. We express this as the (inverse-closed) deletion request $\mathcal{U}_A^- = \{B_1(\mathsf{c}), A_1(\mathsf{c})\}$. It is easy to see that a source deletion translating $\mathcal{U}_A^-$ is $\mathcal{U}_D^- = \{r_1(\mathsf{c}), r_2(\mathsf{c})\}$, which has as side-effect also the deletion of $B_2(\mathsf{c})$ from $cl_{\mathcal{T}_0}(\mathcal{M}_0(D_0))$. However, if $\mathcal{T}_0$ contained also the inclusion assertion $A_2 \sqsubseteq B_2$, then $\mathcal{U}_D^-$ would have no side-effect. ◁

In *DL-Lite*$_R$, disjointness assertions might lead to inconsistency, but do not contribute to the implication of ABox assertions [7]. Moreover, inclusion assertions of the

form $B \sqsubseteq \exists R$ directly[3] imply only facts that involve an existentially quantified object, hence they do not directly contribute to the closure of an ABox. Therefore, to account for implied insertions and deletions of ABox facts, we can compute (in polynomial time) the deductive closure $cl(\mathcal{T})$ of the TBox $\mathcal{T}$, and use only the set of inclusion assertions in $cl(\mathcal{T})$ of the form $A' \sqsubseteq A$ or $\exists R \sqsubseteq A$ (i.e., with an atomic concept on the right-hand side), or $R_1 \sqsubseteq R_2$, which we denote $red(\mathcal{T})$. Considering then that in VKGs we want to account for the side-effects caused by $red(\mathcal{T})$ on the (virtual) ABox generated from a source instance via a mapping $\mathcal{M}$, we can do so while compiling away $red(\mathcal{T})$, and using instead of $\mathcal{M}$ the *saturated mapping* $sat_{red(\mathcal{T})}(\mathcal{M})$ [21]. Based on these considerations, we assume from now on that mappings are already saturated, and w.l.o.g. we restrict the attention to the case where the TBox contains only disjointness assertions, i.e., $\mathcal{T} = \mathcal{T}^-$.

We now characterize the translation of ABox deletions and insertions. For deletions, we can rely on the fact that they do not cause violations of disjointness assertions.

Definition 1 (Deletion translation). *Given a VKG instance $\mathcal{J} = \langle\langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, D\rangle$ and a set $\mathcal{U}_A^-$ of ABox deletions, a source update $\mathcal{U}_D^-$ is a* translation of $\mathcal{U}_A^-$ in $\mathcal{J}$ *if* $\mathcal{U}_A^- \cap \mathcal{M}(D \setminus \mathcal{U}_D^-) = \emptyset$. $\mathcal{U}_D^-$ *is an* exact translation *if* $\mathcal{M}(D) \setminus \mathcal{U}_A^- = \mathcal{M}(D \setminus \mathcal{U}_D^-)$. ◁

Hence, an exact translation of $\mathcal{U}_A^-$ in $\mathcal{J}$ ensures that exactly all assertions of $\mathcal{U}_A^-$ are deleted from the (virtual) ABox generated by the mapping from the data sourcex, while a non-exact translation might also cause the deletion of additional assertions not in $\mathcal{U}_A^-$.

Example 2. Consider the VKG specification $\mathcal{P}_1 = \langle \emptyset, \mathcal{M}_1, \mathcal{S}_1\rangle$, where $\mathcal{S}_1 = \{RS/2, SG/2, RG/2\}$, and $\mathcal{M}_1$ consists of the mapping assertion in Eq. (1) and the assertion $RG(x, z) \rightsquigarrow access(x, z)$. Let $\mathcal{J}_1 = \langle \mathcal{P}_1, D_1\rangle$ be a VKG instance, where D_1 consist of the following source tables (we have used names of components for clarity):

RS :

researcher	*supervisor*
r1	s1
r1	s2
r2	s2

SG :

supervisor	*grant*
s1	g1
s2	g1
s1	g2

RG :

researcher	*grant*
r1	g1

We obtain the ABox $\mathcal{M}_1(D_1) = \{access(\texttt{s1}, \texttt{g1}), access(\texttt{r2}, \texttt{g1}), access(\texttt{r1}, \texttt{g2})\}$. Consider now the ABox deletion $\mathcal{U}_A^- = \{access(\texttt{r1}, \texttt{g1})\}$. Then, $\mathcal{U}_D^- = \{RS(\texttt{r1}, \texttt{s2}), SG(\texttt{s1}, \texttt{g1}), RG(\texttt{r1}, \texttt{g1})\}$ is an exact translation of $\mathcal{U}_A^-$ in $\mathcal{J}_1$. Instead, $\mathcal{U}_D^- = \{RS(\texttt{r1}, \texttt{s1}), SG(\texttt{s1}, \texttt{g1}), RS(\texttt{r1}, \texttt{s2}), SG(\texttt{s2}, \texttt{g1}), RG(\texttt{r1}, \texttt{g1})\}$ is a translation of $\mathcal{U}_A^-$ in $\mathcal{J}_1$ but not an exact one, because it deletes from the ABox also $access(\texttt{r1}, \texttt{g2})$. ◁

For insertions, we have ruled out a priori insertions of ABox facts that are inconsistent with $\mathcal{T}$ (hence with $\mathcal{T}^-$), but we still have to consider that side effects might cause an inconsistency.

Definition 2 (Insertion translation). *Given a VKG instance $\mathcal{J} = \langle\langle \mathcal{T}^-, \mathcal{M}, \mathcal{S}\rangle, D\rangle$ and a set $\mathcal{U}_A^+$ of ABox insertions, a source update $\mathcal{U}_D^+$ is a* translation of $\mathcal{U}_A^+$ in $\mathcal{J}$ *if* $\mathcal{M}(D) \cup \mathcal{U}_A^+ \subseteq \mathcal{M}(D \cup \mathcal{U}_D^+)$ *and* $\langle \mathcal{T}^-, \mathcal{M}(D \cup \mathcal{U}_D^+)\rangle$ *is consistent.* $\mathcal{U}_D^+$ *is an* exact translation *of* $\mathcal{U}_A^+$ *if* $\mathcal{M}(D) \cup \mathcal{U}_A^+ = \mathcal{M}(D \cup \mathcal{U}_D^+)$. ◁

[3] $B \sqsubseteq \exists R$ might indirectly imply a proper ABox fact, e.g., for $\mathcal{T} = \{A_1 \sqsubseteq \exists R, \exists R \sqsubseteq A_2\}$ and $\mathcal{A} = \{A_1(\texttt{c})\}$, we have that $A_2(\texttt{c}) \in cl_{\mathcal{T}}(\mathcal{A})$.

Hence, an exact translation of $\mathcal{U}_A^+$ in $\mathcal{J}$ ensures that exactly all assertions of $\mathcal{U}_A^+$ are inserted in the (virtual) ABox generated by the mapping from the data source, while a translation $\mathcal{U}_D^+$ that is non-exact might also cause the insertion of additional assertions not in $\mathcal{U}_A^+$. If the additional assertions cause an inconsistency w.r.t. $\mathcal{T}^-$, we rule out $\mathcal{U}_D^+$.

Example 3. Consider $\mathcal{J}_2 = \langle\langle\emptyset, \mathcal{M}_2, \mathcal{S}_1\rangle, D_2\rangle$, where $\mathcal{M}_2$ consists of the mapping assertion of Eq. (1) and D_2 contains the single tuple $RS(\texttt{r1}, \texttt{s1})$, and the ABox insertion $\mathcal{U}_A^+ = \{access(\texttt{r2}, \texttt{g2})\}$. Then $\mathcal{U}_D^+ = \{RS(\texttt{r2}, \texttt{s2}), SG(\texttt{s2}, \texttt{g2})\}$ is an exact translation of $\mathcal{U}_A^+$ in $\mathcal{J}_2$. Instead, $\mathcal{U}_D^+ = \{RS(\texttt{r2}, \texttt{s1}), SG(\texttt{s1}, \texttt{g2})\}$ is a translation of $\mathcal{U}_A^+$, but not an exact one, because it also leads to the insertion of $access(\texttt{r1}, \texttt{g2})$ in the ABox. ◁

ABox insertions and deletions can have more than one translation over the data source, which is mainly due to the *information loss* caused by VKG mappings. Moreover, the application of a translation over the source may lead to a side-effect due to unwanted deletions or insertions in the ABox. Our aim is to find the translations that minimize the side-effects, ideally exact translations without side effects.

4 Schema Mapping Recovery in VKGs

In our approach to VKG update, we propose to rely on a reverse mapping that maps the TBox back to the source schema. To define such reverse mapping, we make use of the notion of mapping recovery introduced in [2], but suitably adapted to our setting.

For an so-mapping $\mathcal{M}$, from a source schema $\mathcal{S}$ to a TBox $\mathcal{T}$, a *reverse mapping* describes a novel mapping now going from $\mathcal{T}$ back to $\mathcal{S}$, which we call *ontology-to-source mapping* (*os-mapping*). We consider such a mapping $\widehat{\mathcal{M}}$ as a set of pairs $(\mathcal{A}, D)$, with $\mathcal{A}$ an ABox for $\mathcal{T}$ and D a DB instance of $\mathcal{S}$, called *solution* of $\mathcal{A}$ under $\widehat{\mathcal{M}}$. Moreover, we define $\widehat{\mathcal{M}}(\mathcal{A}) = \{D \mid (\mathcal{A}, D) \in \widehat{\mathcal{M}}\}$. Notice that, while an so-mapping $\mathcal{M}$ generates from a DB instance D a unique ABox $\mathcal{M}(D)$ (hence, $\mathcal{M}$ is a function by definition), a reverse mapping $\widehat{\mathcal{M}}$, in general, is not a function but a relation that associates to an ABox a set of DB instances. This is because an so-mapping is in general not injective, i.e., it might map two different instances of $\mathcal{S}$ to the same ABox.

We are interested in os-mappings that maintain semantic consistency with the relationship established by $\mathcal{M}$, and which intuitively represent the inverse of such relationship, in line with the notion of *inverse mapping* for the relational setting [16].

Definition 3 (Recovery of a mapping [2]). *Let $\mathcal{P} = \langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ be a VKG specification and $\widehat{\mathcal{M}}$ an os-mapping. We say $\widehat{\mathcal{M}}$ is a* recovery of $\mathcal{M}$*, if $D \in \widehat{\mathcal{M}}(\mathcal{M}(D))$ for every source instance D of $\mathcal{S}$.* ◁

For a source instance D, if $\widehat{\mathcal{M}}$ is a recovery of a mapping $\mathcal{M}$, then the smaller the set of DB instances provided by $\widehat{\mathcal{M}}(\mathcal{M}(D))$, the more informative is $\widehat{\mathcal{M}}$ about D. This leads to the definition of *maximum recovery*, which intuitively is among the best options to bring the exchanged data back.

Definition 4 (Maximum recovery of a mapping [2]). *Let $\mathcal{P} = \langle\mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ be a VKG specification. A recovery $\widehat{\mathcal{M}}$ of $\mathcal{M}$ is a* maximum recovery*, if for every recovery $\widehat{\mathcal{M}}'$ of $\mathcal{M}$, we have that $\widehat{\mathcal{M}}(\mathcal{M}(D)) \subseteq \widehat{\mathcal{M}}'(\mathcal{M}(D))$.* ◁

Although VKG mappings do not allow for existential variables in the target, IRI-templates (which account for *skolem terms*) give VKG mappings the expressive power to simulate st-tgds that contain existential variables in the target [13,17]. In fact, IRI-templates can also capture plain SO-tgds, which are a form of mappings that always admit a maximum recovery [1]. In [1], an algorithm called PolySOInverse is proposed that, given a schema mapping $\mathcal{M}$ specified as plain SO-tgds, computes a SO-tgd mapping $\widehat{\mathcal{M}}$ that represents the maximum recovery of $\mathcal{M}$, by rewriting each atom in the target part of the SO-tgds as a query over the source. We make use of an adaptation of that algorithm to our setting, which we describe next. To do so, we make the simplifying assumption[4] that all mapping assertions for a concept or role E (including those introduced with mapping saturation) make use of the *same IRI-template*.

Given an n-ary tuple $\boldsymbol{t}(\boldsymbol{x}) = (t_1(\boldsymbol{x}), \ldots, t_n(\boldsymbol{x}))$ of terms over variables $\boldsymbol{x}$, $\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}$ is an n-tuple $(u_1, \ldots, u_n)$ of variables such that if $t_i(\boldsymbol{x}) = t_j(\boldsymbol{x})$, then $u_i = u_j$, otherwise $u_i \neq u_j$. We then define $V_{\boldsymbol{t}(\boldsymbol{x})} = V^1 \wedge \cdots \wedge V^n$, where each V^i is obtained as follows: *(a)* if $t_i(\boldsymbol{x})$ is a variable y in $\boldsymbol{x}$, then V^i is the equality $y = u_i$, *(b)* otherwise, if $t_i(\boldsymbol{x})$ is an IRI-template $f(y_1, \ldots, y_k)$, where each y_j is a variable in $\boldsymbol{x}$, then V^i is $(y_1{=}f^1(u_i)) \wedge \cdots \wedge (y_k{=}f^k(u_i))$, where each f^j is the j-th inverse template of f. We are now ready to define, given an so-mapping $\mathcal{M}$, a specific os-mapping $\widehat{\mathcal{M}}^{mr}$.

Definition 5 (MR-os-mapping). *Let $\mathcal{M} = \bigcup_{E \text{ in } \mathcal{T}} \mathcal{M}_E$ be an so-mapping, where $\mathcal{M}_E = \bigcup_{\ell=1}^{k} \{\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \rightsquigarrow E(\boldsymbol{t}(\boldsymbol{x}))\}$ are all mapping assertions with concept or role E in the target (where $\boldsymbol{t}(\boldsymbol{x})$ is the tuple of IRI-templates specific for E). Then, the set of os-mappings $\widehat{\mathcal{M}}^{mr} = \bigcup_{E \text{ in } \mathcal{T}} \left\{ E(\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}) \rightsquigarrow \bigvee_{\ell=1}^{k} \exists \boldsymbol{x}.\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \wedge V_{\boldsymbol{t}(\boldsymbol{x})} \right\}$ is called the* MR-os-mapping of $\mathcal{M}$. ◁

Example 4. Consider the VKG mapping $\mathcal{M} = \{\exists z.r(x, y, z) \rightsquigarrow P(f(x, y), x)\}$. Then we have $\boldsymbol{t}(x, y, z) = (f(x, y), x)$, and we obtain $\boldsymbol{u}_{\boldsymbol{t}(x,y,z)} = (u_1, u_2)$. Hence $\widehat{\mathcal{M}}^{mr}$ consists of the os-mapping assertion $P(u_1, u_2) \rightsquigarrow \exists x.\exists y.\exists z.r(x, y, z) \wedge x = f^1(u_1) \wedge y = f^2(u_1) \wedge x = u_2$. Notice that we can simplify this os-mapping assertion into the equivalent form $P(u_1, u_2) \rightsquigarrow \exists z.r(f^1(u_1), f^2(u_1), w) \wedge f^1(u_1) = u_2$. ◁

The following result can be shown by following the same line of proof as the analogous result for plain SO-tgds in [1].

Theorem 6. *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification. The os-mapping $\widehat{\mathcal{M}}^{mr}$ is a maximum recovery of $\mathcal{M}$.*

In the following, we restrict the attention to the maximum recovery of a VKG mapping $\mathcal{M}$ constructed as the MR-os-mapping $\widehat{\mathcal{M}}^{mr}$ of $\mathcal{M}$. In fact, we exploit the construction in the proof of the following result.

Theorem 7. *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\widehat{\mathcal{M}}^{mr}$ the MR-os-mapping of $\mathcal{M}$, and D an instance of $\mathcal{S}$. Then, for every $f \in \mathcal{M}(D)$, we have $f \in \mathcal{M}(\widehat{\mathcal{M}}^{mr}(\{f\}))$.*

[4] Such assumption is quite restrictive in practice, but it can be lifted by resorting to the original version of the PolySOInverse algorithm in [1]. We will address this in future work.

5 Data Lineage in VKGs

Data lineage (a.k.a. *provenance*) is concerned with identifying and managing the origin of data in a data management system, and has been studied extensively in the relational setting [9,12]. We now adapt the notions of lineage and *exclusive lineage* to virtual ABox assertions in the VKG setting. Specifically, we are interested in the subsets of the data source tuples that generate a fact in the virtual ABox.

Definition 8 (Lineage). *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\langle \mathcal{P}, D \rangle$ a VKG instance, and $f \in \mathcal{M}(D)$ an ABox assertion. A subset $B \subseteq D$ is a* lineage branch of f *if $f \in \mathcal{M}(B)$, and for every $B' \subsetneq B$, $f \notin \mathcal{M}(B')$. The* lineage of f*, denoted* $\mathsf{lineage}(f, \mathcal{P}, D)$*, is the set of all lineage branches of f.* ◁

Example 5 (Continued from Example 2). For $f = access(\mathtt{r1}, \mathtt{g1})$, $\mathsf{lineage}(f, \mathcal{P}, D) = \{\{RS(\mathtt{r1}, \mathtt{s1}), SG(\mathtt{s1}, \mathtt{g1})\}, \{RS(\mathtt{r1}, \mathtt{s2}), SG(\mathtt{s2}, \mathtt{g1})\}, \{RG(\mathtt{r1}, \mathtt{g1})\}\}$ ◁

We are also interested in the subsets of the data source tuples that contribute to generating an ABox fact f but do not contribute to any ABox fact other than f.

Definition 9 (Exclusive lineage). *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\langle \mathcal{P}, D \rangle$ a VKG instance, and $f \in \mathcal{M}(D)$ an ABox assertion. A subset B^* of a lineage branch of f is an* exclusive lineage branch of f *if for every $f' \in \mathcal{M}(D) \setminus \{f\}$ and every $B \in \mathsf{lineage}(f', \mathcal{P}, D)$, we have that $B^* \cap B = \emptyset$. The* exclusive lineage of f*, denoted* $\mathsf{elineage}(f, \mathcal{P}, D)$*, is the set of all exclusive lineage branches of f.* ◁

Example 6 (Continued from Example 5). The exclusive lineage of the assertion f is $\mathsf{elineage}(f, \mathcal{P}, D) = \{\{SG(\mathtt{s1}, \mathtt{g1})\}, \{RS(\mathtt{r1}, \mathtt{s2})\}, \{RG(\mathtt{r1}, \mathtt{g1})\}\}$. Tuples $RS(\mathtt{r1}, \mathtt{s1})$ and $SG(\mathtt{s2}, \mathtt{g1})$ from f's lineage are not in f's exclusive lineage because $RS(\mathtt{r1}, \mathtt{s1})$ is also in a lineage branch of $access(\mathtt{r1}, \mathtt{g2})$ and $SG(\mathtt{s2}, \mathtt{g1})$ in one of $access(\mathtt{r2}, \mathtt{g1})$. ◁

Note that an exclusive lineage branch of an ABox fact f is in general *not* a lineage branch of f, i.e., its facts might not generate f via the mapping. Finding minimal subsets of the source instance D that generate via the VKG mapping $\mathcal{M}$ an ABox fact f is in general challenging. We now show that we can use a maximum recovery of $\mathcal{M}$ to guide that search.

Definition 10 ($\widehat{\mathcal{M}}$-instance recovery). *Let $\mathcal{J} = \langle \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle, D \rangle$ be a VKG instance and $\widehat{\mathcal{M}}$ a maximum recovery of $\mathcal{M}$. The* $\widehat{\mathcal{M}}$-instance recovery for D *is the function $\widehat{\mathcal{M}}_D$ from $\mathcal{M}(D)$ to 2^{2^D} such that*

$$\widehat{\mathcal{M}}_D(f) = \{B \mid B \subseteq D, B \in \widehat{\mathcal{M}}(\{f\}), \text{ and for every } B' \subsetneq B, B' \notin \widehat{\mathcal{M}}(\{f\})\}.$$

◁

Notice that, based on the definition and properties of maximum recovery as shown in Sect. 4, for two distinct maximum recoveries $\widehat{\mathcal{M}}^1$ and $\widehat{\mathcal{M}}^2$ and a source instance D, it holds that $\widehat{\mathcal{M}}^1_D = \widehat{\mathcal{M}}^2_D$. This actually follows from the next result.

Proposition 11. *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\langle \mathcal{P}, D \rangle$ a VKG instance, $\widehat{\mathcal{M}}^{mr}$ the MR-os-mapping of $\mathcal{M}$, and $\widehat{\mathcal{M}}^{mr}_D$ the $\widehat{\mathcal{M}}^{mr}$-instance recovery for D. Then, for every ABox assertions $f \in \mathcal{M}(D)$, we have that $\widehat{\mathcal{M}}^{mr}_D(f) = \mathsf{lineage}(f, \mathcal{P}, D)$.*

Proof. Let $f = E(\boldsymbol{a})$ for some concept or role E and some tuple $\boldsymbol{a}$ of ground terms. We recall that, by Theorem 6, $\widehat{\mathcal{M}}^{mr}$ is a maximum recovery of $\mathcal{M}$.

($\subseteq$) Let $B \in \widehat{\mathcal{M}}^{mr}_D(f)$, we need to show that $B \in \mathsf{lineage}(f, \mathcal{P}, D)$. By definition of $\widehat{\mathcal{M}}^{mr}_D(f)$, we have that $B \subseteq D$. Let $E(\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}) \rightsquigarrow \alpha(\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})})$ be the (unique) os-mapping assertion in $\widehat{\mathcal{M}}^{mr}$ for concept/role E, where $\alpha(\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}) = \bigvee_{\ell=1}^{k} \exists \boldsymbol{x}.\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \wedge V_{\boldsymbol{t}(\boldsymbol{x})}$ is a UCQ over $\mathcal{S}$ by construction. Since $f = E(\boldsymbol{a})$ and $B \in \widehat{\mathcal{M}}^{mr}(\{f\})$, we have that $B \models \alpha(\boldsymbol{a})$, which means that there is an $\ell \in [1..k]$ such that $B \models \beta(\boldsymbol{a})$, where $\beta(\boldsymbol{a}) = \exists \boldsymbol{x}.\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \wedge (V_{\boldsymbol{t}(\boldsymbol{x})}[\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}/\boldsymbol{a}])$ and $V_{\boldsymbol{t}(\boldsymbol{x})}[\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}/\boldsymbol{a}]$ is the formula obtained from $V_{\boldsymbol{t}(\boldsymbol{x})}$ by instantiating $\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}$ with $\boldsymbol{a}$. By the form of $\beta(\boldsymbol{a})$, there exists a tuple $\boldsymbol{b}$ of constants such that $\boldsymbol{b} = \hat{\boldsymbol{t}}(\boldsymbol{a})$, where $\hat{\boldsymbol{t}}$ is the sequence of inverse templates corresponding to the IRI-templates $\boldsymbol{t}(\boldsymbol{x})$, such that $B \models \exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{b}, \boldsymbol{w}_\ell)$. From the construction of $\widehat{\mathcal{M}}^{mr}$, we have that $\mathcal{M}$ contains the mapping assertion $\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \rightsquigarrow E(\boldsymbol{t}(\boldsymbol{x}))$, and since $B \models \exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{a}, \boldsymbol{w}_\ell)$, we have that $f \in \mathcal{M}(B)$. Moreover, by the definition of $\widehat{\mathcal{M}}$-instance recovery, there is no $B' \subsetneq B$ such that $B' \models \exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{a}, \boldsymbol{w}_\ell)$. Hence, $B \in \mathsf{lineage}(f, \mathcal{P}, D)$.

($\supseteq$) Let $B \in \mathsf{lineage}(f, \mathcal{P}, D)$, we need to show that $B \in \widehat{\mathcal{M}}^{mr}_D(f)$. From Definition 8 we know that $f \in \mathcal{M}(B)$. Hence there is a mapping assertion $\exists \boldsymbol{w}.\beta(\boldsymbol{x}, \boldsymbol{w}) \rightsquigarrow E(\boldsymbol{t}(\boldsymbol{x})) \in \mathcal{M}$ and a tuple $\boldsymbol{b}$ of constants such that $\boldsymbol{b} = \hat{\boldsymbol{t}}(\boldsymbol{a})$, where $\hat{\boldsymbol{t}}$ is the sequence of inverse templates corresponding to the IRI-templates $\boldsymbol{t}(\boldsymbol{x})$, such that $B \models \exists \boldsymbol{w}.\beta(\boldsymbol{b}, \boldsymbol{w})$. Moreover, for every $B' \subsetneq B$, since by the definition of lineage $f \notin \mathcal{M}(B')$, we have that $B' \not\models \exists \boldsymbol{w}.\beta(\boldsymbol{b}, \boldsymbol{w})$. From the definition and construction of $\widehat{\mathcal{M}}^{mr}$, we have that $E(\boldsymbol{u}_{\boldsymbol{t}(\boldsymbol{x})}) \rightsquigarrow \bigvee_{\ell=1}^{k} \exists \boldsymbol{x}.\exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) \wedge V_{\boldsymbol{t}(\boldsymbol{x})} \in \widehat{\mathcal{M}}^{mr}$, where for some $\ell \in [1..k]$, $\varphi_\ell(\boldsymbol{x}, \boldsymbol{w}_\ell) = \beta(\boldsymbol{x}, \boldsymbol{w})$. Hence, we have $B \models \exists \boldsymbol{w}_\ell.\varphi_\ell(\boldsymbol{b}, \boldsymbol{w}_\ell)$, which entails that $B \in \widehat{\mathcal{M}}^{mr}(\{f\})$ and $B' \notin \widehat{\mathcal{M}}^{mr}(\{f\})$. Since by the definition of lineage, $B \subseteq D$, we conclude that $B \in \widehat{\mathcal{M}}^{mr}_D(f)$. □

6 Update Framework in VKGs

When the translated ABox updates are propagated back through the mappings they might produce side effects, and we are interested in minimizing them, where we consider as measure the set difference between the desired and the obtained ABoxes. We use the function $\mathit{flatten}(X) = \bigcup_{S \in X} S$, which flattens a set X of sets into a set.

6.1 ABox Deletions

We first consider ABox deletions. We formalize the relationship between data lineage and translation of ABox deletions, and based on that, we provide an algorithm that translates ABox deletions into suitable source deletions.

Example 7 (Continued from Example 2). The set $\mathcal{U}^-_D = \mathit{flatten}(\mathsf{lineage}(f, \mathcal{P}, D))$ of source tuples is a translation of $\mathcal{U}^-_A = \{f\}$ but is not an exact translation. Instead, $\mathcal{U}^-_D = \mathit{flatten}(\mathsf{elineage}(f, \mathcal{P}, D))$ is an exact translation of $\mathcal{U}^-_A$. ◁

Algorithm 1: TRANSLATEDELETION

input : A VKG instance $\mathcal{J} = \langle \mathcal{P}, D \rangle$ with $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$.
A set $\mathcal{U}_A^-$ of ABox deletions.
output: A set $\mathbf{U}_D^-$ of translations over the source of the set $\mathcal{U}_A^-$.

$B^* \leftarrow \mathit{flatten}(\bigcup_{f \in \mathcal{U}_A^-} \mathsf{elineage}(f, \mathcal{P}, D))$; *// Compute flattening of elineage*
$D' \leftarrow D \setminus B^*$; *// Remove from D tuples in the exclusive lineage*
if $\mathcal{M}(D) \setminus \mathcal{U}_A^- = \mathcal{M}(D')$ **then return** $\mathbf{U}_D^- = B^*$;
$\mathbf{B} = \{B_1, \ldots, B_n\} \leftarrow \bigcup_{f \in \mathcal{U}_A^-} \mathsf{lineage}(f, \mathcal{P}, D')$; *// Compute lineage branches in D'*
$\mathbf{S} \leftarrow \{\{t_1, \ldots, t_n\} \mid t_i \in B_i, \text{for } i \in [1..n]\}$;
$\mathbf{U}_D^- \leftarrow \{\}$;
for each translation $T^* \in \mathbf{S}$ **do**
  $add \leftarrow true$;
  for each translation $T \in \mathbf{U}_D^-$ **do**
    if $\text{COMPARE}_{\mathcal{J}}^-(T \cup B^*, T^* \cup B^*) =$ '<' **then** $add \leftarrow false$
    else if $\text{COMPARE}_{\mathcal{J}}^-(T^* \cup B^*, T \cup B^*) =$ '<' **then** $\mathbf{U}_D^- \leftarrow \mathbf{U}_D^- \setminus \{T\}$
  if add **then** $\mathbf{U}_D^- \leftarrow \mathbf{U}_D^- \cup \{T^*\}$
return $\{T \cup B^* \mid T \in \mathbf{U}_D^-\}$

We observe that, in principle, since no proper subset of a lineage branch of an assertion f generates f under the VKG mapping, we can delete f from the ABox by simply removing one tuple from each lineage branch of f, and we should do so while ensuring a minimal side-effect. E.g., in Example 7, by deleting one tuple per lineage branch of f, we can obtain the set $T = \{RS(\texttt{r1}, \texttt{s2}), SG(\texttt{s1}, \texttt{g1})\ RG(\texttt{r1}, \texttt{g1})\}$, which is a translation of $\mathcal{U}_A^- = \{f\}$ over the source data (which in this case has no side effect).

Proposition 12. *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\langle \mathcal{P}, D \rangle$ a VKG instance, $f \in \mathcal{M}(D)$ such that $\mathsf{lineage}(f, \mathcal{P}, D) = \{B_1, \ldots, B_n\}$, and $\mathbf{T} = \{\{t_1, \ldots, t_n\} \mid t_i \in B_i\}$. Then for every $T \in \mathbf{T}$, we have that $f \notin \mathcal{M}(D \setminus T)$.*

Proof. Let $T = \{t_1, \ldots, t_n\} \in \mathbf{T}$. We have to show that $f \notin \mathcal{M}(D \setminus T)$. From the definition of lineage for an ABox assertion f, it follows that, for every $i \in [1..n]$ and $B_i' = B_i \setminus \{t_i\}$, we have that $f \notin \mathcal{M}(B_i')$. This means that, by removing t_i from D, the branch B_i will not be anymore in $\mathsf{lineage}(f, \mathcal{P}, D \setminus \{t_i\})$. Therefore, by removing T from D, $\mathsf{lineage}(f, \mathcal{P}, D \setminus T)$ will be empty, hence $f \notin \mathcal{M}(D \setminus T)$. □

From Definition 6, we can see that none of the source tuples contained in the exclusive lineage of an ABox assertion contribute to generating via the mapping any other assertion in the ABox, which means that their deletion is guaranteed to be side-effect free. Unfortunately, as observed also in [8], deleting from the source the exclusive lineage of a given ABox assertion will not always lead to its deletion in the ABox.

We propose the algorithm TRANSLATEDELETION, which takes as input a VKG instance $\mathcal{J}$ and a set $\mathcal{U}_A^-$ of ABox deletions. It first computes the exclusive lineage branch list $\mathbf{B}^*$ of every assertion in $\mathcal{U}_A^-$ and checks whether deleting the tuples in $\mathbf{B}^*$ will delete $\mathcal{U}_A^-$ from the ABox. If so, it returns the tuples in $\mathbf{B}^*$ as an exact translation. Instead, if some assertions of $\mathcal{U}_A^-$ still persist in the ABox, the algorithm searches for

translations of $\mathcal{U}_A^-$ with minimum side effects by considering one tuple per branch in the remaining lineage list. However, the notion of *right* translation is not unique, as it depends on the metric used to evaluate the distance between the updated ABox and the desired one. In this regard, we use an abstract function called $\text{COMPARE}_{\mathcal{J}}^-$ that compares two translations T_1 and T_2 in terms of their side-effect and returns either '=' or '<', if T_1 has equal, resp., less side-effect than T_2 based on the metric under consideration. Our algorithm returns the set of translations with a minimum side effect.

The number of iterations in the search loop is bounded by $k^{|\mathbf{B}|}$ where $\mathbf{B} = \bigcup_{f \in \mathcal{U}_A^-} \mathsf{lineage}(f, \mathcal{P}, D')$ and $\mathcal{U}_A^-$ in $D' = D \setminus \mathit{flatten}(\bigcup_{f \in \mathcal{U}_A^-} \mathsf{elineage}(f, \mathcal{P}, D))$ (i.e., D after removing the exclusive lineage of all tuples in $\mathcal{U}_A^-$) and k is the size of the largest branch of $\mathbf{B}$. Note that, in general, we expect k and $|\mathbf{B}|$ to be small.

6.2 ABox Insertions

Inserting new ABox assertions is more challenging than deletions because, for these new assertions, we do not yet have a lineage in the database that we can manipulate. Also, the existential variables in source queries of mappings can result in infinitely many possible source insertions. Similar to the translation of ABox deletions, we want to minimize side effects in the ABox.

Example 8 (Continued from Example 3). From $\mathcal{M}$, we get the MR-os-mapping $\widehat{\mathcal{M}}^{mr} = \{access(\mathtt{r}, \mathtt{g}) \rightarrow \exists w.RS(\mathtt{r}, w) \wedge SG(w, \mathtt{g})\}$, which is a maximum recovery of $\mathcal{M}$. Then, for $\mathcal{U}_A^+ = \{access(\mathtt{r2}, \mathtt{g2})\}$, we have that $\widehat{\mathcal{M}}^{mr}(\mathcal{U}_A^+) = \{\exists w.RS(\mathtt{r2}, w) \wedge SG(w, \mathtt{g2})\}$. However, the source update is incompletely specified because of the existential variable w. By analyzing our data source D, we see that if we assign to w the value $\mathtt{s2}$, we obtain the source insertion $\{RS(\mathtt{r2}, \mathtt{s2}), SG(\mathtt{s2}, \mathtt{g2})\}$, which is an exact translation of $\mathcal{U}_A^+$. If we assign to w any value other than $\mathtt{s2}$, we still get a translation (but not an exact translation). ◁

This example shows that the choice of the assignment to the existential variables in the translated insertion request plays a crucial role in minimizing the side effects in the ABox. Also, since an ABox insertion can have several translations over the source, one has to find the one that, with a proper assignment, leads to a minimal side effect.

Towards computing an optimal translation of an ABox insertion request, let us first provide the definition of insertion branch and insertion tree of a set of ABox insertions. For convenience, we make use of γ, defined as $\gamma(\bigwedge_{i=1}^{n} r_i(\boldsymbol{a_i})) = \bigcup_{i=1}^{n} \{r_i(\boldsymbol{a_i})\}$, to transform a conjunction of facts into a set of tuples.

Definition 13 (Insertion tree). *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a VKG specification, $\langle \mathcal{P}, D \rangle$ a VKG instance, $\widehat{\mathcal{M}}^{mr}$ the MR-os-mapping of $\mathcal{M}$, and $\mathcal{U}_A^+ = \{f_1, \ldots, f_n\}$ an ABox insertion. Then $B = \bigcup_{i=1}^{n} \{\exists \boldsymbol{w}_i.\varphi_i(\boldsymbol{a}_i, \boldsymbol{w}_i)\}$ is an* insertion branch of $\mathcal{U}_A^+$ for $\widehat{\mathcal{M}}^{mr}$ *if for $i \in [1..n]$ we have that $\gamma(\varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i))) \in \widehat{\mathcal{M}}^{mr}(\{f_i\})$, where $\boldsymbol{w}_i$ is the set of existential variables in $\exists \boldsymbol{w}_i.\varphi_i(\boldsymbol{x}_i, \boldsymbol{w}_i)$, $\boldsymbol{a}_i$ is a tuple of ground terms instantiating all free variables in $\exists \boldsymbol{w}_i.\varphi_i(\boldsymbol{x}_i, \boldsymbol{w}_i)$, and η_i is an arbitrary assignment of constants in $\mathbf{N}_I$ to $\boldsymbol{w}_i$. The* insertion tree of $\mathcal{U}_A^+$ for $\widehat{\mathcal{M}}^{mr}$ *is the set of its insertion branches.* ◁

Algorithm 2: TRANSLATEINSERTION

input: A VKG instance $\mathcal{J} = \langle\langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle, D\rangle$.
A set $\mathcal{U}_A^+$ of ABox insertions.
output: A set $\mathbf{U}_D^+$ of translations over the source of the set $\mathcal{U}_A^+$.

Compute the MR-os-mapping $\widehat{\mathcal{M}}^{mr}$ of $\mathcal{M}$;
Compute the insertion tree $\mathbf{B} = \{B_1, \ldots, B_n\}$ of $\mathcal{U}_A^+$ for $\widehat{\mathcal{M}}^{mr}$;
$\mathbf{U}_D^+ \leftarrow \{\}$;
for each branch $B_i \in \mathbf{B}$ of the form $\bigcup_{i=1}^{n}\{\exists \boldsymbol{w}_i.\varphi_i(\boldsymbol{a}_i, \boldsymbol{w}_i)\}$ **do**
  $\Delta_{\boldsymbol{w}_i} \leftarrow \{b_1, \ldots, b_{|\boldsymbol{w}_i|}\} \cup \Delta_A \cup \Delta_D$, where Δ_A and Δ_D are the constants in $\mathcal{U}_A^+$ and D respectively, and each b_i is a fresh value from $\mathbf{N}_I$ not in $\Delta_A \cup \Delta_D$;
  for each assignment $\eta(\boldsymbol{w}_i) \subseteq \Delta_{\boldsymbol{w}_i}$ **do**
    $T^* \leftarrow \bigcup_{i=1}^{n} \gamma(\varphi_i(\boldsymbol{a}_i, \eta(\boldsymbol{w}_i)))$;
    $add \leftarrow true$;
    for each translation $T \in \mathbf{U}_D^+$ **do**
      if $\text{COMPARE}_{\mathcal{J}}^{+}(T, T^*) =$ '<' **then** $add \leftarrow false$
      else if $\text{COMPARE}_{\mathcal{J}}^{+}(T^*, T) =$ '<' **then** $\mathbf{U}_D^+ \leftarrow \mathbf{U}_D^+ \setminus \{T\}$
    if add **then** $\mathbf{U}_D^+ \leftarrow \mathbf{U}_D^+ \cup \{T^*\}$
return $\mathbf{U}_D^+$

Example 9 (Continued from Example 8). The insertion tree of the ABox insertion $\mathcal{U}_A^+ = \{access(\texttt{r2}, \texttt{g2})\}$ is $\mathbf{B} = \{\{\exists w.RS(\texttt{r2}, w) \wedge SG(w, \texttt{g2})\}\}$. ◁

We now show that, for every instantiation of its existential variables with constants in $\mathbf{N}_I$, an insertion branch provides a set of source tuples that generates through the mapping a set of ABox assertions that includes $\mathcal{U}_A^+$.

Theorem 14. *Let $\mathcal{P} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ be a VKG specification, $\langle \mathcal{P}, D\rangle$ a VKG instance, $\widehat{\mathcal{M}}^{mr}$ the MR-os-mapping of $\mathcal{M}$, $\mathcal{U}_A^+ = \{f_1, \ldots, f_n\}$, $\bigcup_{i=1}^{n}\{\exists \boldsymbol{w}_i.\varphi_i(\boldsymbol{a}_i, \boldsymbol{w}_i)\}$ an insertion branch of $\mathcal{U}_A^+$ for $\widehat{\mathcal{M}}^{mr}$, and η_i an assignment to $\boldsymbol{w}_i$, for $i \in [1..n]$. Then, the source update $\mathcal{U}_D^+ = \bigcup_{i=1}^{n} \gamma(\varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i)))$ is a* translation *of $\mathcal{U}_A^+$.*

Proof. By Theorem 7, we know that $f_i \in \mathcal{M}(\widehat{\mathcal{M}}^{mr}(\{f_i\}))$, and since we have $\gamma(\varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i))) \in \widehat{\mathcal{M}}^{mr}(\{f_i\})$, we obtain $f_i \in \mathcal{M}(\gamma(\varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i))))$. By combining all the assertions f_i in $\mathcal{U}_A^+$, we obtain $\mathcal{U}_A^+ \subseteq \bigcup_{i=1}^{n} \mathcal{M}(\varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i)))$, hence $\mathcal{U}_A^+ \subseteq \mathcal{M}(\bigcup_{i=1}^{n} \varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i)))$, since $\mathcal{M}$ is specified as UCQs over the source and UCQs are monotone queries. Again, by the monotonicity of UCQs, we finally obtain $\mathcal{U}_A^+ \cup \mathcal{M}(D) \subseteq \mathcal{M}(\bigcup_{i=1}^{n} \varphi_i(\boldsymbol{a}_i, \eta_i(\boldsymbol{w}_i)) \cup D)$. □

This result shows that applying $\widehat{\mathcal{M}}^{mr}$ to an ABox insertion request always leads to a translation in the source data. Hence, we need to compute an insertion branch and an assignment to its existential variables that leads to a minimal side-effect. We propose the algorithm TRANSLATEINSERTION, which takes as input a VKG instance $\mathcal{J}$ and a set $\mathcal{U}_A^+$ of ABox insertions, computes the insertion tree of $\mathcal{U}_A^+$, and for every branch B in the tree, assigns to each of its existential variables (if it has any) a constant that might be from the source instance, from the insertion request, or fresh. In some situations,

assigning arbitrary values to these variables, known as "do not care" variables, may be viable, allowing the translation process to fill in any value to maintain semantic consistency [12]. Similar to the deletion algorithm, we use $\text{COMPARE}^{+}_{\mathcal{J}}$ to compare insertion translations (resulting from the instantiation of existential variables) in terms of side effects in the ABox, and we keep the ones with minimal side effects. The algorithm needs to compute the insertion tree, and to do so it uses the MR-os-mapping $\widehat{\mathcal{M}}^{mr}$. The size of the insertion tree is $|\mathcal{U}_A^+|^k$, where $k = \max_{f_i \in \mathcal{U}_A^+}(|\widehat{\mathcal{M}}^{mr}(\{f_i\})|)$.

7 Conclusions and Discussion

We have studied an instance-level approach for updates in the context of VKGs through the lens of an ontology, where an update is specified through either a set of ABox deletions or a set of ABox insertions over the ontology. Based on the notion of maximum recovery of a VKG mapping, we have proposed algorithms that compute the set of changes in the source that realize a given ontology-based update with the minimum side effect. The computation is, in the worst case, exponential in the size of the update. For ABox deletions, the translation is computed at run-time, but our algorithm can compute at compile-time the maximum recovery and the lineage of all assertions in the ABox.

We are currently working on a technique to combine ABox deletions and insertions. We are also interested in studying optimization techniques when searching for possible translations, and in particular in understanding how constraints in the data source can be used to reduce the space of possible translations of ABox updates. We are also planning to implement our techniques in state-of-the-art VKG tools such as Ontop [5].

Acknowledgments. This research has been partially supported by the Province of Bolzano and DFG through the project D2G2 (DFG grant n. 500249124), by the HEU project CyclOps (grant agreement n. 101135513), and by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation.

References

1. Arenas, M., Pérez, J., Reutter, J., Riveros, C.: The language of plain SO-tgds: composition, inversion and structural properties. JCSS **79**(6), 763–784 (2013). https://doi.org/10.1016/j.jcss.2013.01.002
2. Arenas, M., Pérez, J., Riveros, C.: The recovery of a schema mapping: Bringing exchanged data back. ACM TODS **34**(4), 1–48 (2009). https://doi.org/10.1145/1620585.1620589
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003). https://doi.org/10.1017/CBO9780511711787
4. Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM TODS **6**(4), 557–575 (1981). https://doi.org/10.1145/319628.319634
5. Calvanese, D., et al.: Ontop: answering SPARQL queries over relational databases. Semantic Web J. **8**(3), 471–487 (2017). https://doi.org/10.3233/SW-160217
6. Calvanese, D., et al.: Ontologies and databases: the DL-Lite approach. In: Tessaris, S., et al. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03754-2_7

7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. JAR **39**, 385–429 (2007). https://doi.org/10.1007/s10817-007-9078-x
8. Cui, Y., Widom, J.: Run-time translation of view tuple deletions using data lineage. Tech. rep., Stanford University (2001). http://ilpubs.stanford.edu:8090/496/1/2001-24.pdf
9. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. VLDBJ **12**(1), 41–58 (2003). https://doi.org/10.1007/s00778-002-0083-8
10. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. ACM TODS **25**(2), 179–227 (2000). https://doi.org/10.1145/357775.357777
11. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recommendation, W3C (2012). http://www.w3.org/TR/r2rml/
12. Dayal, U., Bernstein, P.A.: On the correct translation of update operations on relational views. ACM TODS **7**(3), 381–416 (1982). https://doi.org/10.1145/319732.319740
13. De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Using ontologies for semantic data integration. In: Flesca, S., Greco, S., Masciari, E., Saccà, D. (eds.) A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years. SBD, vol. 31, pp. 187–202. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-61893-7_11
14. De Giacomo, G., Lembo, D., Oriol, X., Savo, D.F., Teniente, E.: Practical update management in ontology-based data access. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 225–242. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_14
15. De Giacomo, G., Oriol, X., Rosati, R., Savo, D.F.: Instance-level update in DL-Lite ontologies through first-order rewriting. JAIR (2021). https://doi.org/10.1613/jair.1.12414
16. Fagin, R.: Inverting schema mappings. ACM TODS **32**(4), 2–53 (2007). https://doi.org/10.1145/1292609.1292615
17. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: second-order dependencies to the rescue. ACM TODS **30**(4), 994–1055 (2005). https://doi.org/10.1145/1114244.1114249
18. Fagin, R., Ullman, J.D., Vardi, M.Y.: On the semantics of updates in databases. In: Proceedings of the PODS, pp. 352–365 (1983). https://doi.org/10.1145/588058.588100
19. Flouris, G.: On belief change in ontology evolution. AI Commun. **19**(4), 395–397 (2006)
20. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In: Proceedings of the KR, pp. 387–394 (1991)
21. Kontchakov, R., Rezk, M., Rodríguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 552–567. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_35
22. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, W3C (2012). http://www.w3.org/TR/owl2-profiles/
23. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics **10**, 133–173 (2008). https://doi.org/10.1007/978-3-540-77688-8_5
24. Winslett, M.: Updating Logical Databases. Cambridge University Press (1990)
25. Xiao, G., et al.: Ontology-based data access: a survey. In: Proceedings of the IJCAI, pp. 5511–5519. IJCAI Org (2018).https://doi.org/10.24963/ijcai.2018/777
26. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: an overview of systems and use cases. Data Intell. (2019). https://doi.org/10.1162/dint_a_00011
27. Zheleznyakov, D., Kharlamov, E., Nutt, W., Calvanese, D.: On expansion and contraction of DL-Lite knowledge bases. J. Web Semantics **57**, 100484 (2019). https://doi.org/10.1016/j.websem.2018.12.002