Planning and Change in Graph Structured Data under Description Logics Constraints [*]

Shqiponja Ahmetaj[1], Diego Calvanese[2], Magdalena Ortiz[1], and Mantas Šimkus[1]

[1] Institute of Information Systems, Vienna University of Technology, Austria
[2] KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Italy

1 Introduction

The complex structure and increasing size of information that has to be managed in today's applications calls for flexible mechanisms for storing such information, making it easily and efficiently accessible, and facilitating its change and evolution over time. The paradigm of *graph structured data* (GSD) [8] has gained popularity recently as an alternative to traditional relational DBs that provides more flexibility and thus can overcome the limitations of an a priori imposed rigid structure on the data. Indeed, differently from relational data, GSD do not require a schema to be fixed a priori. This flexibility makes them well suited for many emerging application areas such as managing Web data, information integration, persistent storage in object-oriented software development, or management of scientific data. Concrete examples of models for GSD are RDFS [4], object-oriented data models, and XML.

Here, we build on recent work that advocates the use of Description Logics (DLs) for managing change in GSD [7] that happens as the result of (agents or users) executing actions. We consider GSD, understood in a broad sense, as information represented by means of a node and edge labeled graph, in which the labels convey semantic information. We identify GSD with the *finite* structures over which DLs are interpreted, and use DL knowledge bases as descriptions of constraints and properties of the data. We express actions using a specially tailored action language in which actions are finite sequences of (possibly conditional) insertions and deletions performed on the extensions of labels. In this setting, the *static verification problem*, which consists on deciding whether the execution of a given action will *preserve* some given integrity constraints on any possible GSD, has been studied in [7]. Here, we discuss further problems that can be considered as variants of planning, such as deciding if there is a sequence of actions that leads a given structure into a state where some property (either desired or not) holds, or deciding whether a given sequence of actions leads every structure into a state where some property necessarily holds. We develop algorithms for variations of these problems, and characterize their computational complexity. We consider both the case of known and arbitrary initial state. We show that existence of a plan (of unbounded length) is undecidable even for lightweight DLs and a simple forms of actions. Motivated by this we study planning for plans of bounded length, and provide tight complexity bounds for the considered variants of the problem. An extended version of this work, with proofs of the technical results and more detailed discussion can be found in [1].

[*] This research has been partially supported by FWF projects T515 and P25518, by WWTF project ICT12-015, by EU IP Project Optique FP7-318338, and by the Wolfgang Pauli Institute.

2 Description Logics for Graph Structured Data

We now introduce the DL $\mathcal{ALCHOIQ}br$, which we use to describe constraints on GSD. In DLs, the domain of interest is modeled using individuals (denoting objects), concepts (denoting sets of objects), and roles (denoting binary relations between objects). We assume countably infinite sets $\mathsf{N_R}$ of *role names*, $\mathsf{N_C}$ of *concept names*, $\mathsf{N_I}$ of *individual names*, and $\mathsf{N_V}$ of *variables*. *Roles* are defined inductively: *(i)* if $p \in \mathsf{N_R}$, then p and p^- (the *inverse* of p) are roles; *(ii)* if $\{t, t'\} \subseteq \mathsf{N_I} \cup \mathsf{N_V}$, then $\{(t_1, t_2)\}$ is a role; *(iii)* if r_1, r_2 are roles, then $r_1 \cup r_2$ and $r_1 \setminus r_2$ are roles; and *(iv)* if r is a role and C is a concept, then $r|_C$ is a role. *Concepts* are defined inductively as well: *(i)* each $A \in \mathsf{N_C}$ is a concept; *(ii)* if $t \in \mathsf{N_I} \cup \mathsf{N_V}$, then $\{t\}$ is a concept (called *nominal*); *(iii)* if C_1, C_2 are concepts, then $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, and $\neg C_1$ are concepts; *(iv)* if r is a role, C is a concept, and n is a non-negative integer, then $\exists r.C$, $\forall r.C$, $\leqslant n\, r.C$, and $\geqslant n\, r.C$ are concepts.

A *concept* (resp., *role*) *inclusion* has the form $\alpha_1 \sqsubseteq \alpha_2$, where α_1, α_2 are concepts (resp., roles). A *concept* (resp., *role*) *assertion* has the form $t : C$ (resp., $(t, t') : r$), where $\{t, t'\} \subseteq \mathsf{N_I} \cup \mathsf{N_V}$, C is a concept, and r is a role. Concepts, roles, inclusions, and assertions without variables are called *ordinary*. We define ($\mathcal{ALCHOIQ}br$-)*formulae* inductively: inclusion and assertions are formulas, and if $\mathcal{K}_1, \mathcal{K}_2$ are formulas, so are $\mathcal{K}_1 \wedge \mathcal{K}_2$, $\mathcal{K}_1 \vee \mathcal{K}_2$, and $\neg\mathcal{K}_1$. A *knowledge base (KB)* is a formula with no variables.

Notice that $\mathcal{ALCHOIQ}br$ extends the expressive DL $\mathcal{ALCHOIQ}$ [3] with Boolean combinations of axioms, a constructor for a singleton role, union, difference and restrictions of roles, and variables as place-holders for individuals. We consider here also the lightweight DL *DL-Lite* [6], which closely matches the expressive power of traditional conceptual modeling formalisms, such as UML class diagrams and ER schemas.

As usual in DLs, the semantics is given in terms of interpretations. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, where $\Delta^\mathcal{I} \neq \emptyset$ is the *domain*, $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ for each $A \in \mathsf{N_C}$, $p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for each $p \in \mathsf{N_R}$, and $o^\mathcal{I} \in \Delta^\mathcal{I}$ for each $o \in \mathsf{N_I}$. We make the *unique name assumption (UNA)*, i.e., distinct individuals are interpreted as distinct objects. For ordinary roles $\{(o_1, o_2)\}$, we let $\{(o_1, o_2)\}^\mathcal{I} = \{(o_1^\mathcal{I}, o_2^\mathcal{I})\}$, and for ordinary roles $r|_C$, we let $(r|_C)^\mathcal{I} = \{(e_1, e_2) \mid (e_1, e_2) \in r^\mathcal{I} \text{ and } e_2 \in C^\mathcal{I}\}$. The function $\cdot^\mathcal{I}$ is extended to the remaining ordinary concepts and roles in the usual way [3].

An interpretation $\mathcal{I}$ *satisfies* an ordinary inclusion $\alpha_1 \sqsubseteq \alpha_2$, denoted $\mathcal{I} \models \alpha_1 \sqsubseteq \alpha_2$, if $\alpha_1^\mathcal{I} \subseteq \alpha_2^\mathcal{I}$, and an ordinary assertion $\beta = o : C$ (resp., $\beta = (o_1, o_2) : r$), denoted $\mathcal{I} \models \beta$, if $o^\mathcal{I} \in C^\mathcal{I}$ (resp., $(o_1^\mathcal{I}, o_2^\mathcal{I}) \in r^\mathcal{I}$). Satisfaction is extended to knowledge bases as follows: *(i)* $\mathcal{I} \models \mathcal{K}_1 \wedge \mathcal{K}_2$ if $\mathcal{I} \models \mathcal{K}_1$ and $\mathcal{I} \models \mathcal{K}_2$; *(ii)* $\mathcal{I} \models \mathcal{K}_1 \vee \mathcal{K}_2$ if $\mathcal{I} \models \mathcal{K}_1$ or $\mathcal{I} \models \mathcal{K}_2$; *(iii)* $\mathcal{I} \models \neg\mathcal{K}$ if $\mathcal{I} \not\models \mathcal{K}$. If $\mathcal{I} \models \mathcal{K}$, then $\mathcal{I}$ is a *model* of $\mathcal{K}$. The *finite satisfiability problem* is to decide given a KB $\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ with $\Delta^\mathcal{I}$ finite. The finite satisfiability problem for $\mathcal{ALCHOIQ}br$ KBs has the same computational complexity as for the standard $\mathcal{ALCHOIQ}$:

We are interested in the problem of *effectively* managing GSDs satisfying the constraints expressed in a DL KB $\mathcal{K}$. Hence, we must assume that such data are of *finite* size, i.e., they correspond naturally to *finite interpretations that satisfy the constraints* in $\mathcal{K}$. In other words, we consider configurations of the GSD that are finite models of $\mathcal{K}$.

Many of the reasoning problems we study here will be reduced to finite satisfiability, which is NExpTime-complete for $\mathcal{ALCHOIQ}br$ [7]. Contrast this with *DL-Lite*, for which (finite) satisfiability is NLogSpace-complete [2].

Example 1. The following interpretation $\mathcal{I}_1$ represents (part of) the project database of a research institute. There are two active projects and three employees working in them.

$$\begin{array}{lll} \mathsf{Prj}^{\mathcal{I}_1} = \{p_1, p_2\}, & \mathsf{ActivePrj}^{\mathcal{I}_1} = \{p_1, p_2\}, & \mathsf{FinishedPrj}^{\mathcal{I}_1} = \{\}, \\ \mathsf{Empl}^{\mathcal{I}_1} = \{e_1, e_3, e_7\}, & \mathsf{worksFor}^{\mathcal{I}_1} = \{(e_1, p_1), (e_3, p_1), (e_7, p_2)\}. & \end{array}$$

We assume constants $\mathsf{p_i}$ with $\mathsf{p_i}^{\mathcal{I}} = p_i$ for projects, and analogously constants $\mathsf{e_i}$ for employees. The KB $\mathcal{K}_1$ expresses constraints on this project database: all projects are active or finished, the domain of worksFor are the employees, and its range the projects.

$$(\mathsf{Prj} \sqsubseteq \mathsf{ActivePrj} \sqcup \mathsf{FinishedPrj}) \wedge (\exists \mathsf{worksFor}.\top \sqsubseteq \mathsf{Empl}) \wedge (\exists \mathsf{worksFor}^{-}.\top \sqsubseteq \mathsf{Prj})$$

3 Updating Graph Structured Data

For manipulating GSD, we use the following language. The basic actions allow one to insert or delete individuals from extensions of concepts, and pairs of individuals from extensions of roles. The candidates for additions and deletions can be chosen by means of complex concepts and roles. The language also allows for composition of actions and conditional action execution.

Definition 1 (Action language). Basic actions β *and* (complex) actions α *are built according to the following grammar, where A is a concept name, C is an arbitrary concept, p is a role name, r is an arbitrary role, and $\mathcal{K}$ is an arbitrary $\mathcal{ALCHOIQ}br$–formula. The special symbol ε denotes the* empty action*:*

$$\beta \longrightarrow (A \oplus C) \mid (A \ominus C) \mid (p \oplus r) \mid (p \ominus r) \qquad\qquad \alpha \longrightarrow \beta \cdot \alpha \mid \mathcal{K}\,?\,\alpha;\alpha \mid \varepsilon$$

A (complex) action α is called simple *if (i) no (concept or role) inclusions occur in α, and (ii) all concepts of α are Boolean combinations of concept names, nominals, and concepts of the form $\exists r.\top$.*

A substitution *is a function σ from $\mathsf{N_V}$ to $\mathsf{N_I}$. For a formula, an action or an action sequence Γ, we use $\sigma(\Gamma)$ to denote the result of replacing in Γ every occurrence of a variable x by the individual $\sigma(x)$. An action α is* ground *if it has no variables. An action α' is called a* ground instance *of an action α if $\alpha' = \sigma(\alpha)$ for some substitution σ.*

Intuitively, an application of an action $(A \oplus C)$ on an interpretation $\mathcal{I}$ stands for the addition of the content of $C^{\mathcal{I}}$ to $A^{\mathcal{I}}$. In turn, removing $C^{\mathcal{I}}$ from $A^{\mathcal{I}}$ can be done by applying $(A \ominus C)$ on $\mathcal{I}$. The two operations can also be performed on extensions of roles. Composition stands for successive action execution, and a conditional action $\mathcal{K}\,?\,\alpha_1;\alpha_2$ says that α_1 is executed if the interpretation is a model of $\mathcal{K}$, and α_2 is executed otherwise. We now formally define the semantics of actions.

Definition 2. *Assume an interpretation $\mathcal{I}$ and let E be a concept or role name. If E is a concept, let $W \subseteq \Delta^{\mathcal{I}}$, and if E is a role, let $W \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Then let $\mathcal{I} \oplus_E W$ (resp., $\mathcal{I} \ominus_E W$) denote the interpretation $\mathcal{I}'$ such that (i) $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$, (ii) $E^{\mathcal{I}'} = E^{\mathcal{I}} \cup W$*

(resp., $E^{\mathcal{I}'} = E^{\mathcal{I}} \setminus W$), and (iii) $E_1^{\mathcal{I}'} = E_1^{\mathcal{I}}$, for all symbols $E_1 \neq E$. For a ground action α, we define a mapping S_α from interpretations to interpretations:

$$
\begin{array}{ll}
S_{(A \oplus C)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \oplus_A C^{\mathcal{I}}) & S_{(p \oplus r)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \oplus_p r^{\mathcal{I}}) \\
S_{(A \ominus C)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \ominus_A C^{\mathcal{I}}) & S_{(p \ominus r)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \ominus_p r^{\mathcal{I}}) \\
S_\varepsilon(\mathcal{I}) = \mathcal{I} & S_{\mathcal{K}?\alpha_1;\alpha_2}(\mathcal{I}) = \begin{cases} S_{\alpha_1}(\mathcal{I}), & \text{if } \mathcal{I} \models \mathcal{K}, \\ S_{\alpha_2}(\mathcal{I}), & \text{if } \mathcal{I} \not\models \mathcal{K}. \end{cases}
\end{array}
$$

Note that we have not defined the semantics of actions with variables. In our approach, all variables of an action are seen as parameters whose values are given before execution by a substitution with actual individuals, i.e., by grounding.

The *static verification problem* amounts to checking, given a KB $\mathcal{K}$ and an action α, that for every finite model $\mathcal{I}$ of $\mathcal{K}$ and every ground instance α' of α, $S_{\alpha'}(\mathcal{I}) \models \mathcal{K}$, i.e., the execution of α preserves the satisfaction of the constraints expressed by $\mathcal{K}$.

Theorem 1 ([7]). *The static verification problem is co*NEXPTIME*-complete for input KBs in $\mathcal{ALCHOIQ}br$, and co*NP*-complete for DL-Lite input KBs and simple actions.*

The proof of this theorem relies on a *regression* technique that incorporates into a given $\mathcal{K}$ the effects of an action α, and reduces reasoning about its effects on any structure satisfying $\mathcal{K}$ to reasoning about a single KB. In particular, static verification is reduced to finite KB unsatisfiability. This regression technique is also the main tool for most upper bounds in the next section, but we must omit proofs due to lack of space.

4 Planning

When data evolves, there may be desirable states that we want to ensure, or undesirable states that we want to avoid. For example, a finished project should never be made active again. We tackle such issues next, through some *planning* problems. We use DLs to describe states of KBs, which may act as goals or preconditions. A *plan* is a sequence of actions from a given set, whose execution leads from the current state to a state that satisfies a given goal. To support unbounded introduction of fresh values in the data, we allow for the domain to be expanded with a finite set of domain elements.

Definition 3. *Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be a finite interpretation, Act a finite set of actions, and $\mathcal{K}$ a KB (the* goal *KB). A finite sequence $P = \langle \alpha_1, \ldots, \alpha_n \rangle$ of ground instances of actions from Act is called a* plan for $\mathcal{K}$ from $\mathcal{I}$ *(of* length n*), if there exists a finite set Δ with $\Delta^{\mathcal{I}} \cap \Delta = \emptyset$ such that $S_{\alpha_1 \cdots \alpha_n}(\mathcal{I}') \models \mathcal{K}$, where $\mathcal{I}' = \langle \Delta^{\mathcal{I}} \cup \Delta, \cdot^{\mathcal{I}} \rangle$.*

Example 2. Recall $\mathcal{I}_1$ and $\mathcal{K}_1$ from Example 1. The following *goal* KB requires that p_1 is not an active project, and that e_1 is an employee. Consider the following actions α_1 and α_2. Action α_1 moves p_1 from the active to the finished projects, and removes the employees working only for p_1 from the corresponding tables. Action α_2 transfers e_1 from project p_1 to project p_2 (only if the necessary preliminary checks are successful).

$$
\begin{aligned}
\mathcal{K}_g &= \neg(\mathsf{p}_1{:}\mathsf{ActivePrj}) \wedge \mathsf{e}_1{:}\mathsf{Empl} \\
\alpha_1 &= \mathsf{ActivePrj} \ominus \{\mathsf{p}_1\} \cdot \mathsf{FinishedPrj} \oplus \{\mathsf{p}_1\} \cdot \mathsf{worksFor} \ominus \mathsf{worksFor}|_{\{\mathsf{p}_1\}} \cdot \\
&\qquad \mathsf{Empl} \ominus \neg\exists \mathsf{worksFor}.\mathsf{Prj} \\
\alpha_2 &= (\mathsf{p}_2{:}\mathsf{Prj} \wedge (\mathsf{e}_1, \mathsf{p}_1){:}\mathsf{worksFor}) \mathbin{?} \mathsf{worksFor} \ominus \{(\mathsf{e}_1, \mathsf{p}_1)\} \cdot \mathsf{worksFor} \oplus \{(\mathsf{e}_1, \mathsf{p}_2)\};\varepsilon
\end{aligned}
$$

The sequence $\langle \alpha_2, \alpha_1 \rangle$ is a plan for $\mathcal{K}_g$ from $\mathcal{I}_1$, and the interpretation $S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)$ that reflects the resulting status of the data looks as follows. Note that $S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1) \models \mathcal{K}_1 \wedge \mathcal{K}_g$.

$$\mathsf{Prj}^{S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)} = \{p_1, p_2\}, \quad \mathsf{ActivePrj}^{S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)} = \{p_2\}, \quad \mathsf{FinishedPrj}^{S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)} = \{p_1\},$$
$$\mathsf{Empl}^{S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)} = \{e_1, e_7\}, \quad \mathsf{worksFor}^{S_{\alpha_2 \cdot \alpha_1}(\mathcal{I}_1)} = \{(e_1, p_2), (e_7, p_2)\},$$

We define the next planning problems:

(P1) Given a set Act of actions, a finite interpretation $\mathcal{I}$, and a goal KB $\mathcal{K}$, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$?

(P2) Given a set Act of actions and a pair $\mathcal{K}_{pre}$, $\mathcal{K}$ of formulae, does there exist a substitution σ and a plan for $\sigma(\mathcal{K})$ from some finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$?

(P1) is the classic plan existence problem, formulated in the setting of GSD. (P2) also aims at deciding plan existence, but rather than the full actual state of the data, we have as an input a *precondition* KB, and we are interested in deciding the existence of a plan from some of its models. To see the relevance of (P2), consider the complementary problem: a 'no' instance of (P2) means that, from every relevant initial state, (undesired) goals cannot be reached. For instance, $\mathcal{K}_{pre} = \mathcal{K}_{ic} \wedge x : \mathsf{FinishedPrj}$ and $\mathcal{K} = x : \mathsf{ActivePrj}$ may be used to check whether starting with GSD that satisfies the integrity constraints and contains some finished project p, it is possible to make p an active project again.

Unfortunately, these problems are undecidable in general.

Theorem 2. *(P1) and (P2) are undecidable, already for DL-Lite KBs and simple actions.*

To regain decidability, we define 'bounded' versions of these problems. (P1) becomes decidable if the size of Δ in Definition 3 is bounded. (P2) remains undecidable even for $\Delta = \emptyset$, but it becomes decidable if we place a bound on the length of plans. In the following we assume numbers are coded in unary.

($\mathrm{P1_b}$) Given a set Act of actions, a finite interpretation $\mathcal{I}$, a goal KB $\mathcal{K}$, and a positive integer k, does there exist a plan for $\mathcal{K}$ from $\mathcal{I}$ where $|\Delta| \leq k$?

($\mathrm{P2_b}$) Given a set of actions Act, a pair $\mathcal{K}_{pre}, \mathcal{K}$ of formulae, and a positive integer k, does there exist a substitution σ and a plan of length at most k for $\sigma(\mathcal{K})$ from some finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$?

The problem ($\mathrm{P1_b}$) can be solved in polynomial space for $\mathcal{ALCHOIQ}br$, and a matching lower bound holds even for settings more restricted than *DL-Lite* (the goal is a basic assertion $a : C$ and only simple actions with no complex DL expressions are allowed). Note that planning in our setting is not harder than deciding plan existence in standard automated planning formalisms such as propositional STRIPS [5].

Theorem 3. *The problem ($P1_b$) is* PSPACE*-complete.*

Now we establish the complexity of ($\mathrm{P2_b}$), both in the general setting where $\mathcal{K}_{pre}$ and $\mathcal{K}$ are in $\mathcal{ALCHOIQ}br$, and for the restricted case of *DL-Lite*.

Theorem 4. *The problem ($P2_b$) is* NEXPTIME*-complete. It is* NP*-complete if $\mathcal{K}_{pre}, \mathcal{K}$ are expressed in DL-Lite and all actions in Act are simple.*

Now we consider problems that are related to ensuring that plans *always* achieve a goal $\mathcal{K}$, given a possibly incomplete description $\mathcal{K}_{pre}$ of the initial data. They are variants of the so-called *conformant* planning, which deals with incomplete information. The first such problem is to 'certify' that a candidate plan is always a plan for the goal.

(C) Given a sequence P of actions and formulae $\mathcal{K}_{pre}, \mathcal{K}$, is $\sigma(P)$ a plan for $\sigma(\mathcal{K})$ from every finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

Finally, we are interested in deciding the existence of a plan that always achieves the goal, for every possible state satisfying the precondition. Solving this problem corresponds to the automated *synthesis* of a program for reaching a certain condition.

(S) Given a set Act of actions and formulae $\mathcal{K}_{pre}, \mathcal{K}$, does there exist a sequence P of actions from Act such that $\sigma(P)$ is a plan for $\sigma(\mathcal{K})$ from every finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

($\mathrm{S_b}$) Given a set Act of actions, formulae $\mathcal{K}_{pre}, \mathcal{K}$, and a positive integer k, does there exist a sequence P of actions from Act such that $\sigma(P)$ is a plan for $\sigma(\mathcal{K})$ of length at most k, from every finite $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every substitution σ?

Theorem 5. *(i) Problem (S) is undecidable, already for DL-Lite KBs and simple actions. (ii) Problems (C) and (S_b) are co*NExpTime*-complete. (iii) If $\mathcal{K}_{pre}, \mathcal{K}$ are expressed in DL-Lite and all actions in Act are simple, then (C) is in co*NP *and (S_b) is in* $\mathrm{NP}^{\mathrm{NP}}$.

The upper bounds in the third item are tight for an extension of *DL-Lite* that allows to use regression for capturing action effects. We omit details due to lack of space.

5 Conclusions

We believe this work provides powerful tools for analyzing the effects of executing complex actions on graph structured data, possibly in the presence of integrity constraints expressed in DLs. The considered problems are intractable even for restricted fragments of *DL-Lite* and forms of actions. However, these are worst case bounds, and we believe that practicable algorithms can still be obtained. In our future work we want to verify this, and identify meaningful restrictions to regain tractability.

References

1. S. Ahmetaj, D. Calvanese, M. Ortiz, and M. Šimkus. Managing change in graph-structured data using description logics. In *Proc. of AAAI*, 2014. Long version with proofs available at `http://arxiv.org/abs/1404.4274`.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *JAIR*, 36:1–69, 2009.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. CUP, 2003.
4. D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, W3C, Feb. 2004. `http://www.w3.org/TR/rdf-schema/`.
5. T. Bylander. The computational complexity of propositional STRIPS planning. *AIJ*, 69:165–204, 1994.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 39(3):385–429, 2007.
7. D. Calvanese, M. Ortiz, and M. Šimkus. Evolving graph databases under description logic constraints. In *Proc. of DL*, volume 1014 of *CEUR,* `ceur-ws.org`, pages 120–131, 2013.
8. S. Sakr and E. Pardede, editors. *Graph Data Management: Techniques and Applications*. IGI Global, 2011.