# Evolving Graph Databases under Description Logic Constraints*

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

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

**Abstract.** In the setting of graph-structured data, description logics are well suited to impose constraints that capture the semantics of the domain of interest. When the data evolves as a result of operations carried out by users or applications, it is important to ensure that the satisfaction of the constraints is preserved, analogously to the consistency requirement for transactions in relational databases. In this paper we introduce a simple action language in which actions are finite sequences of insertions and deletions performed on concept/roles, and we study static verification in this setting. Specifically, we address the problem of verifying whether the constraints are still satisfied in the state resulting from the execution of a given action, for every possible initial state satisfying them. We are able to provide a tight coNExpTime bound for a very expressive DL, and show that the complexity drops to coNP-complete for the case of *DL-Lite*.

## 1 Introduction

*Graph databases* are gaining increasing importance due to the adoption of data formats like RDF, and are fundamental for storing, for example, web data in the form of RDF triplestores, and other forms of semi-structured data. Given the very large amounts of data currently available in these stores, the development of automated management tools for them is becoming a pressing problem. Indeed, many of the crucial aspects of managing classical relational databases are also relevant for graph databases. For instance, as in traditional databases, *integrity constraints* on graph databases are important to capture the semantics of the domain of interest. Databases may evolve as a result of operations carried out by users or applications, and it is important to ensure that this does not result in the constraints being violated. This is essentially the *consistency* property of database transactions. A transaction encapsulates a sequence of modifications to the data, which are executed and committed as a unit. A transaction

* This research has been partially supported by FWF projects T515-N23 and P25518-N23, by WWTF project ICT12-015, by EU IP Project Optique FP7-318338, and by the Wolfgang Pauli Institute.

is consistent if its execution over a legal database state never leads to an illegal one. Verifying the consistency of transactions is a crucial problem that has been studied extensively, for different kinds of transactions and constraints, over traditional relational databases [11], object-oriented databases [2,12,3], and deductive databases [5], to name a few. Most of these works adoptexpressive formalisms like (extensions of) first or higher order predicate logic [3], or undecidable tailored languages [11] to express the constraints and the operations on the data. Systems for performing verification are often implemented using theorem provers, and complete algorithms can not be devised.

In contrast, the setting we consider here is that of graph databases where integrity constraints are expressed in Description Logics (DLs) [1]. DLs are decidable languages that have been strongly advocated for managing data repositories [6], and are particularly natural for talking about graph databases. To express the changes to the databases, we introduce a simple action language in which actions are finite sequences of insertions and deletions performed on unary and binary predicates (concepts and roles, in DL jargon). We consider the *static verification problem* for this setting, that is, the problem of verifying whether the constraints are still satisfied in the state resulting from the execution of a given action, for every possible initial state satisfying them. As usual in DLs, the semantics of the DL knowledge bases expressing the constraints is defined in terms of interpretations. In turn, graph databases can be naturally seen as finite DL interpretations. Given a set of constraints expressed by a knowledge base $\mathcal{K}$, we have that a (graph) database satisfies the constraints in $\mathcal{K}$ iff it is a model of $\mathcal{K}$ when seen as an interpretation. Hence, in this paper we talk about interpretations only, and identify the satisfaction of constraints with the modelhood problem. The updates described by the action language are similar in spirit to the knowledge base (more concretely, ABox) updates studied in other works, e.g., [7], but are done directly on interpretations rather than on (the instance level of) knowledge bases. In this framework, we are able to show that the static verification problem is decidable and provide tight complexity bounds for it, using two different DLs as constraint languages. Specifically, we provide a tight coNExpTime bound for a very expressive DL, and show that the complexity drops to coNP-completeness for the case of a variation of *DL-Lite* [4].

## 2 Description Logic for Constraint Specification

We now define the description logic that we use as a constraint language for graph databases. We call it $\mathcal{ALCHOIQ}br$, which is the standard $\mathcal{ALCHOIQ}$ extended with Boolean combinations of axioms and a constructor for a singleton role. 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 $r \in \mathsf{N_R}$, then $r$ and $r^-$ (the *inverse* of $r$) are roles; *(ii)* if $\{t, t'\} \subseteq \mathsf{N_I} \cup \mathsf{N_V}$, then $\{(t_1, t_2)\}$ is also a role; *(iii)* if $r_1, r_2$ are roles, then $r_1 \cup r_2$, and $r_1 \setminus r_2$ are also roles.

*Concepts* are defined inductively as well: *(i)* if $A \in \mathsf{N_C}$, then $A$ 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 also 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 also concepts.

A *concept* (resp., *role*) *inclusion* is an expression of the form $\alpha_1 \sqsubseteq \alpha_2$, where $\alpha_1, \alpha_2$ are concepts (resp., roles). Expressions of the form $t : C$ and $(t, t') : r$, where $\{t, t'\} \subseteq \mathsf{N_I} \cup \mathsf{N_V}$, $C$ is a concept, and $r$ is a role, are called *concept assertions* and *role assertions*, respectively. Concepts, roles, inclusions and assertions that have no variables are called *ordinary*. The role of variables will become apparent in Section 3. We define ($\mathcal{ALCHOIQ}br$-)*formulae* inductively: *(i)* every inclusion and assertion is a formula; *(ii)* if $\mathcal{K}_1, \mathcal{K}_2$ are formulas, then so are $\mathcal{K}_1 \wedge \mathcal{K}_2$, $\mathcal{K}_1 \vee \mathcal{K}_2$ and $\dot{\neg}\mathcal{K}_1$. If a formula $\mathcal{K}$ has no variables, it is called a *knowledge base (KB)*.

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 $A \in \mathsf{N_C}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $r \in \mathsf{N_R}$, and $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $o \in \mathsf{N_I}$. For the ordinary roles of the form $\{(o_1, o_2)\}$, we let $\{(o_1, o_2)\}^{\mathcal{I}} = \{(o_1^{\mathcal{I}}, o_2^{\mathcal{I}})\}$. The function $\cdot^{\mathcal{I}}$ is extended to the remaining ordinary concepts and roles in the usual way. Assume an interpretation $\mathcal{I}$. For an ordinary inclusion $\alpha_1 \sqsubseteq \alpha_2$, $\mathcal{I}$ *satisfies* $\alpha_1 \sqsubseteq \alpha_2$ (in symbols, $\mathcal{I} \models \alpha_1 \sqsubseteq \alpha_2$) if $\alpha_1^{\mathcal{I}} \subseteq \alpha_2^{\mathcal{I}}$. For an ordinary assertion $\beta = o : C$ (resp., $\beta = (o_1, o_2) : r$), $\mathcal{I}$ *satisfies* $\beta$ (in symbols, $\mathcal{I} \models \beta$) if $o^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp., $(o_1^{\mathcal{I}}, o_2^{\mathcal{I}}) \in r^{\mathcal{I}}$). The notion of 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)* $\mathcal{I} \models \dot{\neg}\mathcal{K}$ 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* (resp., *unsatisfiability*) *problem* is to decide given a KB $\mathcal{K}$ if there exists (resp., doesn't exist) 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 $\mathcal{ALCHOIQ}$:

**Theorem 1.** *Finite satisfiability of $\mathcal{ALCHOIQ}br$ KBs is* NExpTime*-complete.*

*Proof (sketch).* The lower bound comes from [13]. The upper bound can be obtained by a straightforward translation into the two variable fragment with counting, for which finite satisfiability is NExpTime-complete [10].

## 3 Actions

We now define an action language that allows us to express finite sequences of operations on interpretations (i.e., graph databases).

**Definition 1 (Action language).** *A* basic action *$\beta$ is given by the following grammar:*

$$\beta \;\longrightarrow\; (A \oplus C) \;\mid\; (A \ominus C) \;\mid\; (r \oplus p) \;\mid\; (r \ominus p),$$

*where $A$ is a concept name, $r$ is a role name, $C$ is an arbitrary concept, and $p$ is an arbitrary role. Then,* (complex) actions *are given by the following grammar:*

$$\alpha \;\longrightarrow\; \beta \cdot \alpha \;\mid\; \mathcal{K}\,?\,\alpha : \alpha \;\mid\; \epsilon,$$

*where $\beta$ is a basic action and $\mathcal{K}$ is an arbitrary $\mathcal{ALCHOIQ}br$–formula. The special symbol $\epsilon$ denotes the* empty action.

*An action $\alpha$ is* ground *if it has no variables. An action $\alpha'$ is called a* ground instance *of an action $\alpha$ if $\alpha'$ is ground and it can be obtained from $\alpha$ by replacing each variable by an individual name from $\mathsf{N_I}$. For an action $\alpha$, we will sometimes write $\alpha(\boldsymbol{x})$, where $\boldsymbol{x}$ is a tuple containing exactly the variables of $\alpha$.*

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. In addition, complex actions allow for composing basic actions and for conditional action execution. In order to formally define the semantics of actions, we first introduce the notion of *interpretation updates*.

**Definition 2 (Interpretation updates).** *Assume an interpretation $\mathcal{I}$ and let $\sigma$ be a concept or role name. If $\sigma$ is a concept, let $W \subseteq \Delta^{\mathcal{I}}$ be a unary relation, otherwise, if $\sigma$ is a role, let $W \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ be a binary relation. Then we let $\mathcal{I} \oplus_\sigma W$ (resp., $\mathcal{I} \ominus_\sigma W$) denote the interpretation $\mathcal{I}'$ such that*

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$,
- $\sigma_1^{\mathcal{I}'} = \sigma_1^{\mathcal{I}}$ *for all symbols* $\sigma_1 \neq \sigma$, *and*
- $\sigma^{\mathcal{I}'} = \sigma^{\mathcal{I}} \cup W$ *(resp.,* $\sigma^{\mathcal{I}'} = \sigma^{\mathcal{I}} \setminus W$*).*

Now we can define the semantics of ground actions inductively as follows.

**Definition 3.** *Given a ground action $\alpha$, we define a mapping $S_\alpha$ from interpretations to interpretations as follows:*

$$S_{(A\oplus C)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \oplus_A C^{\mathcal{I}}) \qquad S_{(r\oplus p)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \oplus_r p^{\mathcal{I}})$$
$$S_{(A\ominus C)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \ominus_A C^{\mathcal{I}}) \qquad S_{(r\ominus p)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \ominus_r p^{\mathcal{I}})$$
$$S_\epsilon(\mathcal{I}) = \mathcal{I} \qquad S_{\mathcal{K}?\alpha_1:\alpha_2}(\mathcal{I}) = \begin{cases} S_{\alpha_1}(\mathcal{I}), & \text{if } \mathcal{I} \models \mathcal{K}, \\ S_{\alpha_2}(\mathcal{I}), & \text{if } \mathcal{I} \not\models \mathcal{K}. \end{cases}$$

*Example 1.* The following interpretation $\mathcal{I}_1$ represents (part of) the project database of some research institute. There are two active projects, and there are four employees of which three work in the active projects.

$$\begin{aligned}
\mathsf{ActiveProject}^{\mathcal{I}_1} &= \{\mathit{P20840}, \mathit{P24090}\},\\
\mathsf{Project}^{\mathcal{I}_1} &= \{\mathit{P20840}, \mathit{P24090}\},\\
\mathsf{ConcludedProject}^{\mathcal{I}_1} &= \{\},\\
\mathsf{Employee}^{\mathcal{I}_1} &= \{\mathit{E01}, \mathit{E03}, \mathit{E04}, \mathit{E07}\},\\
\mathsf{ProjectEmployee}^{\mathcal{I}_1} &= \{\mathit{E01}, \mathit{E03}, \mathit{E07}\},\\
\mathsf{PermanentEmployee}^{\mathcal{I}_1} &= \{\mathit{E04}\},\\
\mathsf{worksFor}^{\mathcal{I}_1} &= \{(\mathit{E01}, \mathit{P20840}), (\mathit{E03}, \mathit{P20840}), (\mathit{E07}, \mathit{P24090})\},\\
\mathsf{P20840}^{\mathcal{I}_1} &= \mathit{P20840},\\
\mathsf{P24090}^{\mathcal{I}_1} &= \mathit{P24090}.
\end{aligned}$$

The following action captures the termination of project P20840, which is removed from the active projects and added to the concluded ones. The employees working for this project are removed from the project employees.

$$\begin{aligned}\alpha_1 = \ & \mathsf{ActiveProject} \ominus \{\mathsf{P20840}\}\cdot \\ & \mathsf{ConcludedProject} \oplus \{\mathsf{P20840}\}\cdot \\ & \mathsf{ProjectEmployee} \ominus \exists \mathsf{worksFor}.\{\mathsf{P20840}\}\end{aligned}$$

The interpretation $S_{\alpha_1}(\mathcal{I}_1)$ that reflects the status of the database after action $\alpha_1$ looks as follows:

$$\begin{aligned}\mathsf{ActiveProject}^{S_{\alpha_1}(\mathcal{I}_1)} &= \{\mathit{P24090}\},\\ \mathsf{Project}^{S_{\alpha_1}(\mathcal{I}_1)} &= \{\mathit{P20840}, \mathit{P24090}\},\\ \mathsf{ConcludedProject}^{S_{\alpha_1}(\mathcal{I}_1)} &= \{\mathit{P20840}\},\\ \mathsf{Employee}^{S_{\alpha_1}(\mathcal{I}_1)} &= \{\mathit{E01}, \mathit{E03}, \mathit{E04}, \mathit{E07}\},\\ \mathsf{ProjectEmployee}^{S_{\alpha_1}(\mathcal{I}_1)} &= \{\mathit{E07}\},\\ \mathsf{PermanentEmployee}^{\mathcal{I}_1} &= \{\mathit{E04}\},\\ \mathsf{worksFor}^{\mathcal{I}_1} &= \{(\mathit{E01}, \mathit{P20840}), (\mathit{E03}, \mathit{P20840}), (\mathit{E07}, \mathit{P24090})\},\\ \mathsf{P20840}^{S_{\alpha_1}(\mathcal{I}_1)} &= \mathit{P20840},\\ \mathsf{P24090}^{S_{\alpha_1}(\mathcal{I}_1)} &= \mathit{P24090}.\end{aligned}$$

In our approach, all the individual variables of an action are seen as parameters, whose values are given before executing an action.

*Example 2.* The following action $\alpha_2$ with variables $x, y, z$ transfers the employee $x$ from project $y$ to project $z$:

$$\begin{aligned}\alpha_2 = \ & (x : \mathsf{Employee} \wedge y : \mathsf{Project} \wedge z : \mathsf{Project} \wedge (x, y) : \mathsf{worksFor})\ ? \\ & \mathsf{worksFor} \ominus \{(x, y)\} \cdot \mathsf{worksFor} \oplus \{(x, z)\} : \epsilon\end{aligned}$$

The action $\alpha_2$ first checks whether $x$ is an employee, $y$ and $z$ are projects, and $x$ works for $y$. If yes, it removes the worksFor link between $x$ and $y$ and creates a worksFor link between $x$ and $z$. If any of the checks fails, it does nothing.

In this paper we focus on *static verification*: we want to know whether, given an action $\alpha$ and a collection of constraints expressed as a KB, the execution of $\alpha$ on $\mathcal{I}$ preserves the satisfaction of the constraints, for any possible finite interpretation $\mathcal{I}$.

**Definition 4 (The static verification problem).** *Let $\mathcal{K}$ be a knowledge base. We say that an action $\alpha$ is* $\mathcal{K}$-preserving *if for every ground instance $\alpha'$ of $\alpha$ and every finite interpretation $\mathcal{I}$, we have that $\mathcal{I} \models \mathcal{K}$ implies $S_{\alpha'}(\mathcal{I}) \models \mathcal{K}$. The* static verification problem *is to decide, given as input an action $\alpha$ and a KB $\mathcal{K}$, whether $\alpha$ is $\mathcal{K}$-preserving.*

*Example 3.* The following KB $\mathcal{K}_1$ expresses constraints on the project database of our running example:

$$\begin{aligned}(\mathsf{Project} \sqsubseteq \mathsf{ActiveProject} \sqcup \mathsf{ConcludedProject}) \wedge \\ (\mathsf{Employee} \sqsubseteq \mathsf{ProjectEmployee} \sqcup \mathsf{PermanentEmployee}) \wedge \\ (\exists \mathsf{worksFor}.\mathsf{Project} \sqsubseteq \mathsf{ProjectEmployee})\end{aligned}$$

The action $\alpha_1$ from Example 1 is not $\mathcal{K}_1$-preserving: $\mathcal{I}_1 \models \mathcal{K}_1$, but $S_{\alpha_1}(\mathcal{I}_1) \not\models \mathcal{K}_1$ since the concept inclusion $\exists \mathsf{worksFor}.\mathsf{Project} \sqsubseteq \mathsf{ProjectEmployee}$ is violated.

## 4 Solving the Static Verification Problem

To obtain an algorithm for the static verification problem we employ a reduction to finite (un)satisfiability of $\mathcal{ALCHOIQ}br$ KBs. In particular, given a knowledge base $\mathcal{K}$ and an action $\alpha$, we build a KB $\mathcal{K}'$ such that $\mathcal{K}'$ is finitely satisfiable iff $\alpha$ is not $\mathcal{K}$-preserving. We start by defining a transformation $\mathsf{TR}_\alpha(\mathcal{K})$ that incorporates an action $\alpha$ into a KB $\mathcal{K}$:

**Definition 5.** *Given a KB $\mathcal{K}$, we use $\mathcal{K}_{L \leftarrow L'}$ to denote the KB that is obtained from $\mathcal{K}$ by replacing every name $L$ by a possibly more complex expression $L'$. Given a KB $\mathcal{K}$ and an action $\alpha$, we define $\mathsf{TR}_\alpha(\mathcal{K})$ as follows:*

*(a)* $\mathsf{TR}_\epsilon(\mathcal{K}) = \mathcal{K}$,
*(b)* $\mathsf{TR}_{(A \oplus C)\cdot\alpha}(\mathcal{K}) = (\mathsf{TR}_\alpha(\mathcal{K}))_{A \leftarrow A \sqcup C}$,
*(c)* $\mathsf{TR}_{(A \ominus C)\cdot\alpha}(\mathcal{K}) = (\mathsf{TR}_\alpha(\mathcal{K}))_{A \leftarrow A \sqcap \neg C}$,
*(d)* $\mathsf{TR}_{(r \oplus p)\cdot\alpha}(\mathcal{K}) = (\mathsf{TR}_\alpha(\mathcal{K}))_{r \leftarrow r \cup p}$,
*(e)* $\mathsf{TR}_{(r \ominus p)\cdot\alpha}(\mathcal{K}) = (\mathsf{TR}_\alpha(\mathcal{K}))_{r \leftarrow r \setminus p}$,
*(f)* $\mathsf{TR}_{(\mathcal{K}_1 ? \alpha_1 : \alpha_2)}(\mathcal{K}) = (\neg\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2}(\mathcal{K}))$

*Example 4.* By applying the transformation above to $\mathcal{K}_1$ and $\alpha_1$, we obtain the following KB $\mathsf{TR}_{\alpha_1}(\mathcal{K}_1)$:

$$\begin{aligned}&(\mathsf{Project} \sqsubseteq (\mathsf{ActiveProject} \sqcap \neg\{\mathsf{P20840}\}) \sqcup (\mathsf{ConcludedProject} \sqcup \{\mathsf{P20840}\})) \wedge \\ &(\mathsf{Employee} \sqsubseteq (\mathsf{ProjectEmployee} \sqcap \neg\exists \mathsf{worksFor}.\{\mathsf{P20840}\}) \sqcup \mathsf{PermanentEmployee}) \wedge \\ &(\exists \mathsf{worksFor}.\mathsf{Project} \sqsubseteq (\mathsf{ProjectEmployee} \sqcap \neg\exists \mathsf{worksFor}.\{\mathsf{P20840}\}))\end{aligned}$$

Note that $\mathcal{I}_1 \not\models \mathsf{TR}_{\alpha_1}(\mathcal{K}_1)$, since $(\exists \mathsf{worksFor}.\mathsf{Project})^{\mathcal{I}_1} = \{\mathit{E01}, \mathit{E03}, \mathit{E07}\}$ and $(\mathsf{ProjectEmployee} \sqcap \neg\exists \mathsf{worksFor}.\{\mathsf{P20840}\})^{\mathcal{I}_1} = \{\mathit{E07}\}$, hence the inclusion

$$\exists \mathsf{worksFor}.\mathsf{Project} \sqsubseteq (\mathsf{ProjectEmployee} \sqcap \neg\exists \mathsf{worksFor}.\{\mathsf{P20840}\}$$

is violated.

We now show that this transformation correctly captures the meaning of ground actions.

**Lemma 1.** *Assume a ground action $\alpha$ and a KB $\mathcal{K}$. For any interpretation $\mathcal{I}$, we have $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$.*

*Proof.* We prove the claim by induction on the structure of $\alpha$. In the base case where $\alpha = \epsilon$, we have $S_\alpha(\mathcal{I}) = \mathcal{I}$ and $\mathsf{TR}_\alpha(\mathcal{K}) = \mathcal{K}$ by definition, and thus the claim holds.

Assume $\alpha = (A \oplus C) \cdot \alpha'$. Let $\mathcal{I}' = S_{(A \oplus C)}(\mathcal{I})$, i.e., $\mathcal{I}'$ coincides with $\mathcal{I}$ except that $A^{\mathcal{I}'} = A^{\mathcal{I}} \cup C^{\mathcal{I}}$. We know that for any KB $\mathcal{K}'$, $\mathcal{I}' \models \mathcal{K}'$ iff $\mathcal{I} \models \mathcal{K}'_{A \leftarrow A \sqcup C}$. In particular, $\mathcal{I}' \models \mathsf{TR}_{\alpha'}(\mathcal{K})$ iff $\mathcal{I} \models (\mathsf{TR}_{\alpha'}(\mathcal{K}))_{A \leftarrow A \sqcup C}$. Since, $(\mathsf{TR}_{\alpha'}(\mathcal{K}))_{A \leftarrow A \sqcup C} = \mathsf{TR}_\alpha(\mathcal{K})$, we get $\mathcal{I}' \models \mathsf{TR}_{\alpha'}(\mathcal{K})$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$. By the induction hypothesis, $\mathcal{I}' \models \mathsf{TR}_{\alpha'}(\mathcal{K})$ iff $S_{\alpha'}(\mathcal{I}') \models \mathcal{K}$. Thus, $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$ iff $S_{\alpha'}(\mathcal{I}') \models \mathcal{K}$. Since $S_{\alpha'}(\mathcal{I}') = S_{\alpha'}(S_{(A \oplus C)}(\mathcal{I})) = S_\alpha(\mathcal{I})$, we have $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$ iff $S_\alpha(\mathcal{I}) \models \mathcal{K}$.

For the cases $\alpha = (A \ominus C) \cdot \alpha'$, $\alpha = (r \oplus p) \cdot \alpha'$, and $\alpha = (r \ominus p) \cdot \alpha'$, the argument is analogous.

Finally, we consider $\alpha = (\mathcal{K}_1 ? \alpha_1 : \alpha_2)$, and assume an arbitrary $\mathcal{I}$. We distinguish two cases:

*(i)* $\mathcal{I} \models \mathcal{K}_1$. Then $S_\alpha(\mathcal{I}) = S_{\alpha_1}(\mathcal{I})$ by definition. By the induction hypothesis, $S_{\alpha_1}(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1}(\mathcal{K})$, hence we have $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1}(\mathcal{K})$. Since $\mathsf{TR}_\alpha(\mathcal{K}) = (\neg\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2}(\mathcal{K}))$, it trivially follows that $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$.

*(ii)* $\mathcal{I} \not\models \mathcal{K}_1$, that is, $\mathcal{I} \models \neg\mathcal{K}_1$. The proof is analogous. We have $S_\alpha(\mathcal{I}) = S_{\alpha_2}(\mathcal{I})$ by definition. By the induction hypothesis, $S_{\alpha_2}(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_2}(\mathcal{K})$, hence we have $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_2}(\mathcal{K})$. Since $\mathsf{TR}_\alpha(\mathcal{K}) = (\neg\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2}(\mathcal{K}))$, it trivially follows that $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$. □

This lemma is the key to our reduction. An action $\alpha$ is not $\mathcal{K}$-preserving iff some model of $\mathcal{K}$ does not model $\mathsf{TR}_{\alpha^*}(\mathcal{K})$, where $\alpha^*$ is a 'canonical' grounding of $\alpha$ obtained by replacing each variable with a fresh individual. This allows us to decide the static verification problem by deciding the satisfiability of $\mathcal{K} \wedge \neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$. Formally, we have:

**Theorem 2.** *Assume a (complex) action $\alpha$ and a KB $\mathcal{K}$. Then the following are equivalent:*

*(i) The action $\alpha$ is not $\mathcal{K}$-preserving.*
*(ii) $\mathcal{K} \wedge \neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable, where $\alpha^*$ is obtained from $\alpha$ by replacing each variable with a fresh individual name not occurring in $\alpha$ and $\mathcal{K}$.*

*Proof. (i)* to *(ii)*. Assume there exist a ground instance $\alpha'$ of $\alpha$ and a finite interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $S_{\alpha'}(\mathcal{I}) \not\models \mathcal{K}$. Then by Lemma 1, $\mathcal{I} \not\models \mathsf{TR}_{\alpha'}(\mathcal{K})$. Thus $\mathcal{I} \models \neg\mathsf{TR}_{\alpha'}(\mathcal{K})$. Suppose $o_1 \to x_1, \ldots, o_n \to x_n$ is the substitution that transforms $\alpha$ into $\alpha'$. Suppose also $o'_1 \to x_1, \ldots, o'_n \to x_n$ is the substitution that transforms $\alpha$ into $\alpha^*$. Take the interpretation $\mathcal{I}^*$ that coincides with $\mathcal{I}$ except for $(o'_i)^{\mathcal{I}^*} = (o_i)^{\mathcal{I}}$. Then $\mathcal{I}^* \models \mathcal{K} \wedge \neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$.

*(ii)* to *(i)*. Assume $\mathcal{K} \wedge \neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable, i.e., there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \not\models \mathsf{TR}_{\alpha^*}(\mathcal{K})$. Then by Lemma 1, $S_{\alpha^*}(\mathcal{I}) \not\models \mathcal{K}$. □

The above reduction leads to the main complexity result of this paper.

**Theorem 3.** *The static verification problem is co*NEXPTIME*-complete in the presence of $\mathcal{ALCHOIQ}br$ KBs.*

*Proof.* The coNEXPTIME upper bound follows from Theorem 2 and the fact that finite satisfiability of $\mathcal{ALCHOIQ}br$ KBs is NEXPTIME-complete (c.f. Theorem 1).

For hardness, we note that finite unsatisfiability of $\mathcal{ALCHOIQ}br$ KBs can be reduced in polynomial time to static verification in the presence of $\mathcal{ALCHOIQ}br$ KBs. Indeed, a KB $\mathcal{K}$ is finitely satisfiable iff $(A' \oplus \{o\})$ is not $(\mathcal{K} \wedge (A \sqsubseteq \neg A') \wedge (o : A))$-preserving, where $A$, $A'$ are fresh concept names and $o$ is a fresh individual. □

## 5 Lowering the Complexity

In this section we consider a restricted setting for which the computational complexity of the static verification problem is lower. We use a variant of *DL-Lite*$_{\mathcal{R}}$ [4], which we call *DL-Lite*$^{+}_{\mathcal{R}}$, as the language for constraint specification. It supports (restricted) Boolean combinations of inclusions and assertions, and allows for more complex concepts and roles in assertions. We also make the *unique name assumption (UNA)*, i.e., for every pair of individuals $o_1$, $o_2$ and interpretation $\mathcal{I}$, we have $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$.

**Definition 6.** *A DL-Lite*$^{+}_{\mathcal{R}}$ *KB $\mathcal{K}$ is a KB satisfying the following conditions:*

- *The operator $\dot{\neg}$ may occur only in front of assertions.*
- *All concept inclusions have the form $C_1 \sqsubseteq C_2$ or $C_1 \sqsubseteq \neg C_2$ of $\mathcal{K}$, with $C_1, C_2 \in \mathsf{N_C} \cup \{\exists r.\top \mid r \in \mathsf{N_R}\} \cup \{\exists r^-.\top \mid r \in \mathsf{N_R}\}$.*
- *All role inclusions have the form $r_1 \sqsubseteq r_2$ or $r_1 \sqsubseteq \neg r_2$, with $r_1, r_2 \in \mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$.*
- *For all concept assertions*[3] *$o : C$ of $\mathcal{K}$, $C \in \mathbf{B}^+$, where $\mathbf{B}^+$ is the smallest set of concepts such that:*
  *(a) $\mathsf{N_C} \subseteq \mathbf{B}^+$,*
  *(b) $\{o'\} \in \mathbf{B}^+$ for all $o' \in \mathsf{N_I}$,*
  *(c) $\exists r.\top \in \mathbf{B}^+$ for all roles $r$,*
  *(d) $\{B_1 \sqcap B_2, B_1 \sqcup B_2, \neg B_1\} \subseteq \mathbf{B}^+$ for all $B_1, B_2 \in \mathbf{B}^+$.*

*A DL-Lite*$_{\mathcal{R}}$ *KB $\mathcal{K}$ is a DL-Lite*$^{+}_{\mathcal{R}}$ *KB that satisfies the following restrictions:*

- *$\mathcal{K}$ is a conjunction of inclusions and assertions,*
- *all assertions in $\mathcal{K}$ are* basic assertions *of the forms $o : C$ with $C \in \mathsf{N_C}$, and $(o, o') : r$ with $r \in \mathsf{N_R}$.*

We also restrict slightly the action language, by allowing only Boolean combinations of (possibly negated) assertions to be used to express the condition $\mathcal{K}$ in actions of the form $\mathcal{K}\,?\,\alpha_1{:}\alpha_2$. This is captured by the notion of *localized* actions.

[3] Note that there is no restriction on role assertion $(o, o') : r$ of $\mathcal{K}$.

**Definition 7.** *If no inclusions occur in an action $\alpha$, then $\alpha$ is called* localized.

*DL-Lite*$_{\mathcal{R}}^{+}$ is expressive enough to allow us to reduce the static verification problem for localized actions to finite unsatisfiability testing.

**Theorem 4.** *The static verification problem for DL-Lite*$_{\mathcal{R}}^{+}$ *KBs and localized actions can be reduced in linear time to finite unsatisfiability testing for DL-Lite*$_{\mathcal{R}}^{+}$ *KBs.*

*Proof.* Assume a *DL-Lite*$_{\mathcal{R}}^{+}$ KB $\mathcal{K}$ and a localized action $\alpha$. Let $\alpha^*$ be the action obtained from $\alpha$ by replacing each variable with a fresh individual name not occurring in $\alpha$ and $\mathcal{K}$. Construct $\mathcal{K}' = \mathcal{K} \wedge \dot{\neg}\mathsf{TR}_{\alpha^*}(\mathcal{K})$. From Theorem 2 we know that $\mathcal{K}'$ is not finitely satisfiable iff $\alpha$ is $\mathcal{K}$-preserving. The KB $\mathcal{K}'$ is not a *DL-Lite*$_{\mathcal{R}}^{+}$ KB, but it can be transformed into an equisatisfiable *DL-Lite*$_{\mathcal{R}}^{+}$ KB in linear time. To this end, turn $\mathcal{K}'$ into negation normal form, i.e., push $\dot{\neg}$ inside so that $\dot{\neg}$ occurs in front of inclusions and assertions only. Then replace every occurrence of $\dot{\neg}(B_1 \sqsubseteq B_2)$ and $\dot{\neg}(r_1 \sqsubseteq r_2)$ in the resulting $\mathcal{K}'$ by $o : B_1 \sqcap \neg B_2$ and $(o, o') : r_1 \setminus r_2$, respectively, where $o, o'$ are fresh individuals. Clearly, the above transformations preserve satisfiability. Moreover, since in $\mathcal{K}$ the operator $\dot{\neg}$ may occur only in front of assertions, and $\alpha$ is localized, every inclusion in the resulting $\mathcal{K}'$ already appears in $\mathcal{K}$. This implies that $\mathcal{K}'$ is a *DL-Lite*$_{\mathcal{R}}^{+}$ KB as desired. □

We next characterize the complexity of finite satisfiability in *DL-Lite*$_{\mathcal{R}}^{+}$.

**Theorem 5.** *Finite satisfiability of DL-Lite*$_{\mathcal{R}}^{+}$ *KBs is* NP*-complete.*

*Proof.* NP-hardness is immediate (e.g., by a reduction from propositional satisfiability). For membership in NP, we define a non-deterministic rewriting procedure that transforms in polynomial time a *DL-Lite*$_{\mathcal{R}}^{+}$ KB into a *DL-Lite*$_{\mathcal{R}}$ KB. In particular, we ensure that a *DL-Lite*$_{\mathcal{R}}^{+}$ KB $\mathcal{K}$ is finitely satisfiable iff there exists a rewriting of $\mathcal{K}$ into a finitely satisfiable *DL-Lite*$_{\mathcal{R}}$ KB. Since satisfiability testing in *DL-Lite*$_{\mathcal{R}}$ is feasible in polynomial time, this yields an NP upper bound for *DL-Lite*$_{\mathcal{R}}^{+}$.

Assume a *DL-Lite*$_{\mathcal{R}}^{+}$ KB $\mathcal{K}$. The rewriting of $\mathcal{K}$ has two steps: first, we get rid of the possible occurrences of $\vee$, and then of the complex concepts and roles in assertions.

For the first step, let $P$ be the set of inclusions and assertions appearing in $\mathcal{K}$. Non-deterministically pick a set $M \subseteq P$ such that $M$ is a model of $\mathcal{K}$, when $\mathcal{K}$ is seen as a propositional formula over $P$. Let $\mathcal{K}_M = \bigwedge_{\alpha \in M} \alpha \wedge \bigwedge_{\alpha' \notin M} \dot{\neg}\alpha'$. Clearly, $\mathcal{K}$ is finitely satisfiable iff we can choose an $M$ with $\mathcal{K}_M$ finitely satisfiable.

In the next step, we show how to obtain from $\mathcal{K}_M$ a *DL-Lite*$_{\mathcal{R}}$ KB. Let $\mathcal{T}$ be the set of inclusions that occur in $\mathcal{K}_M$ and let $\mathcal{A}$ be the set of assertions and their negations occurring in $\mathcal{K}_M$. Recall that the inclusions of $\mathcal{T}$ are inclusions of the standard *DL-Lite*$_{\mathcal{R}}$, but the assertions in $\mathcal{A}$ may contain complex concepts. We non-deterministically complete $\mathcal{A}$ with further assertions to explicate complex concepts and roles. A *completion* of $\mathcal{A}$ is a $\subseteq$-minimal set $\mathcal{A}^+$ of assertions such that:

- $\mathcal{A} \subseteq \mathcal{A}^+$;
- for every assertion $\alpha$, $\alpha \notin \mathcal{A}^+$ or $\dot{\neg}\alpha \notin \mathcal{A}^+$;
- if $o$ is an individual from $\mathcal{K}_M$ and $C_1 \sqsubseteq C_2 \in \mathcal{T}$, then $\dot{\neg}(o : C_1) \in \mathcal{A}^+$ or $o : C_2 \in \mathcal{A}^+$;
- if $(o, o')$ are individuals from $\mathcal{K}_M$ and $r_1 \sqsubseteq r_2 \in \mathcal{T}$, then $\dot{\neg}((o, o') : r_1) \in \mathcal{A}^+$ or $(o, o') : r_2 \in \mathcal{A}^+$;
- if $o : C_1 \sqcap C_2 \in \mathcal{A}^+$, then $o : C_1 \in \mathcal{A}^+$ and $o : C_2 \in \mathcal{A}^+$;
- if $o : C_1 \sqcup C_2 \in \mathcal{A}^+$, then $o : C_1 \in \mathcal{A}^+$ or $o : C_2 \in \mathcal{A}^+$;
- if $o : \exists r.\top \in \mathcal{A}^+$, then $(o, o') : r \in \mathcal{A}^+$ for some fresh $o'$;
- if $o : \neg C \in \mathcal{A}^+$, then $\dot{\neg}(o : C) \in \mathcal{A}^+$;
- if $\dot{\neg}(o : C) \in \mathcal{A}^+$, then $o : \neg C \in \mathcal{A}^+$;
- if $o : \neg\neg C \in \mathcal{A}^+$, then $o : C \in \mathcal{A}^+$;
- if $o : \neg(C_1 \sqcap C_2) \in \mathcal{A}^+$, then $\dot{\neg}(o : C_1) \in \mathcal{A}^+$ or $\dot{\neg}(o : C_2) \in \mathcal{A}^+$;
- if $o : \neg(C_1 \sqcup C_2) \in \mathcal{A}^+$, then $\dot{\neg}(o : C_1) \in \mathcal{A}^+$ and $\dot{\neg}(o : C_2) \in \mathcal{A}^+$;
- if $o : \neg(\exists r.\top) \in \mathcal{A}^+$, then $\dot{\neg}((o, o') : r \in \mathcal{A}^+)$ for all individuals $o'$ of $\mathcal{A}^+$;
- if $(o, o') : r \in \mathcal{A}^+$, then $(o', o) : r^- \in \mathcal{A}^+$;
- if $(o, o') : r_1 \cup r_2 \in \mathcal{A}^+$, then $(o, o') : r_1 \in \mathcal{A}^+$ or $(o, o') : r_2 \in \mathcal{A}^+$;
- if $(o, o') : r_1 \setminus r_2 \in \mathcal{A}^+$, then $(o, o') : r_1 \in \mathcal{A}^+$ and $\dot{\neg}((o, o') : r_2) \in \mathcal{A}^+$;
- if $\dot{\neg}((o, o') : r_1 \cup r_2) \in \mathcal{A}^+$, then $\dot{\neg}((o, o') : r_1) \in \mathcal{A}^+$ and $\dot{\neg}((o, o') : r_2) \in \mathcal{A}^+$;
- if $\dot{\neg}((o, o') : r_1 \setminus r_2) \in \mathcal{A}^+$, then $\dot{\neg}((o, o') : r_1) \in \mathcal{A}^+$ or $(o, o') : r_2 \in \mathcal{A}^+$;
- if $o : \{o'\} \in \mathcal{A}^+$, then $o = o'$;
- if $(o_1, o_2) : \{(o_1', o_2')\} \in \mathcal{A}^+$, then $o_1 = o_1'$ and $o_2 = o_2'$;

Let $\mathcal{A}_b^+$ be the restriction of $\mathcal{A}^+$ to basic assertions. Clearly, $\bigwedge \mathcal{T} \wedge \bigwedge \mathcal{A}_b^+$ is a *DL-Lite*$_\mathcal{R}$ KB. It is not difficult to see that $\mathcal{K}_M$ is finitely satisfiable iff there exists a completion $\mathcal{A}^+$ such that $\bigwedge \mathcal{T} \wedge \bigwedge \mathcal{A}_b^+$ is finitely satisfiable. □

Now we can give a tight bound on the complexity of static verification.

**Theorem 6.** *The static verification problem for DL-Lite*$_\mathcal{R}^+$ *KBs and localized actions is co*NP*-complete.*

*Proof.* The upper bound follows directly from Theorems 4 and 5. The lower bound can be obtained by a reduction from finite unsatisfiability in *DL-Lite*$_\mathcal{R}^+$. To this end, we can employ exactly the same reduction in the proof of Theorem 3. □

It is important to note that coNP-hardness is not only due to the fact that we are using an intractable extension of *DL-Lite*$_\mathcal{R}$ to specify the constraints. Indeed, the given coNP bound is tight even if the constraints are in a very restricted fragment of the core *DL-Lite*: they are a conjunction of disjointness assertions between concept names. The actions needed to show coNP-hardness are also quite restricted: plain sequences of basic concept actions.

**Theorem 7.** *The static verification problem is co*NP*-hard already for KBs of the form* $(A_0 \sqsubseteq \neg A_0') \wedge \ldots (A_n \sqsubseteq \neg A_n')$*, where each* $A_i, A_i'$ *is a concept name, and actions are localized, ground sequences of basic actions of the forms* $(A \oplus C)$ *and* $(A \ominus C)$*.*

*Proof.* We employ the 3-Coloring problem for graphs. Assume a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$. We construct in polynomial time a KB $\mathcal{K}$ and an action $\alpha$ such that $G$ is 3-colorable iff $\alpha$ is not $\mathcal{K}$- preserving. For every $v \in V$, we use 3 concept names $A_v^0, A_v^1, A_v^2$ for the 3 possible colors of the vertex $v$. In addition, we employ a concept name $D$. Let $\mathcal{K}$ be the following KB:

$$\mathcal{K} = (D \sqsubseteq \neg D) \wedge \bigwedge_{(v,v') \in E \wedge 0 \leq c \leq 2} A_v^c \sqsubseteq \neg A_{v'}^c.$$

It remains to define the action $\alpha$. For this we additionally use a nominal $\{o\}$ and fresh concept names $B_1, \ldots, B_n$. We let $\alpha := \alpha_1 \alpha_2^1 \cdots \alpha_2^n \alpha_3$, where

*(i)* $\alpha_1 = (D \oplus \{o\})(B_1 \oplus \{o\}) \cdots (B_n \oplus \{o\})$,
*(ii)* $\alpha_2^i = (B_i \ominus A_i^0)(B_i \ominus A_i^1)(B_i \ominus A_i^3)$ for all $i \in \{1, \ldots, n\}$, and
*(iii)* $\alpha_3 = (D \ominus B_1) \cdots (D \ominus B_n)$.

Assume $\mathcal{I}$ is a model of $\mathcal{K}$ such that $S_\alpha(\mathcal{I}) \not\models \mathcal{K}$. We argue that then $G$ is 3-colorable. Indeed, since $\alpha$ does not modify the extensions of concepts $A_v^c$, $S_\alpha(\mathcal{I}) \not\models \mathcal{K}$ may only hold if $\{o\}^\mathcal{I}$ is in the extension of $D$ after applying $\alpha$ on $\mathcal{I}$. Due to $\alpha_3$, $\{o\}^\mathcal{I}$ is not in the extensions of $B_1, \ldots, B_n$ after applying $\alpha_1 \alpha_2^1 \cdots \alpha_2^n$ on $\mathcal{I}$. Due to $\alpha_1$ and $\alpha_2^n$, for any $i \in \{1, \ldots, n\}$, it must be the case that $\{o\}^\mathcal{I}$ is in the extension of some $A_i^0, A_i^1$ or $A_i^2$ in $\mathcal{I}$. For every $v \in V$, let $col(v) \in \{0, 1, 2\}$ by any value such that $\{o\}^\mathcal{I}$ is in the extension of $A_i^{col(v)}$ in $\mathcal{I}$. Since $\mathcal{I}$ satisfies the disjointness axioms in $\mathcal{K}$, the function *col* is a proper 3-coloring of $G$.

Suppose $G$ is 3-colorable and a proper coloring of $G$ is given by a function $col : V \to \{0, 1, 2\}$. Take any interpretation $\mathcal{I}$ with $\Delta^\mathcal{I} = \{e\}$ and such that (i) $\{o\}^\mathcal{I} = e$, (ii) $D^\mathcal{I} = \emptyset$, (iii) $e \in (A_v^c)^\mathcal{I}$ iff $col(v) = c$. Since *col* is a proper coloring of $G$, $\mathcal{I}$ is a model of $\mathcal{K}$. As easily seen, $S_\alpha(\mathcal{I}) \not\models \mathcal{K}$. □

## 6 Conclusions

In this paper we have studied the static verification problem for evolving graph databases, when the integrity constraints are expressed in DLs and the updates in a simple action language. We have shows that the problem can be reduced to a better known reasoning task: finite satisfiability of knowledge bases. We obtained a tight coNExpTime bound for the problem when the constraints are expressed in a very expressive DL, and a coNP bound if a dialect of *DL-Lite* is considered. This paper is intended to be only a starting point in the study of evolving graph databases under DL constraints, and many challenges remain for future work. For example, the action language we have considered here is rather weak, and a more expressive action language seems desirable. However, many natural extensions, such as 'while' loops in actions, would easily yield an undecidable formalism. Given that the static verification problem is intractable even for weak fragments of the core *DL-Lite* and very restricted forms of actions, it remains to explore how feasible static verification is in practice, and whether there are meaningful restrictions that make the problem tractable. Also other data management reasoning services remain to be considered, including planning, for which we may draw from existing work on DL-based action languages [8,9].

## References


1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. Véronique Benzaken and Xavier Schaefer. Static integrity constraint management in object-oriented database programming languages via predicate transformers. In *Proc. of the Int. Conf. on Object-Oriented Programming (ECOOP'97)*, Lecture Notes in Computer Science, pages 60–84. Springer, 1997.
3. Anthony J. Bonner and Michael Kifer. An overview of Transaction Logic. *Theoretical Computer Science*, 133(2):205–265, 1994.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
5. Robert A. Kowalski, Fariba Sadri, and Paul Soper. Integrity checking in deductive databases. In *Proc. of the 13th Int. Conf. on Very Large Data Bases (VLDB'87)*, pages 61–69, 1987.
6. Maurizio Lenzerini. Ontology-based data management. In *Proc. of the 20th Int. Conf. on Information and Knowledge Management (CIKM 2011)*, pages 5–6, 2011.
7. Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Foundations of instance level updates in expressive description logics. *Artificial Intelligence*, 175(18):2170–2197, 2011.
8. Maja Milicic. Planning in action formalisms based on DLs: First results. In *Proc. of the 20th Int. Workshop on Description Logic (DL 2007)*, volume 250 of *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, 2007.
9. Maja Milicic. *Action, Time and Space in Description Logics*. PhD thesis, TU Dresden, 2008.
10. Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *J. of Logic, Language and Information*, 14(3):369–395, 2005.
11. Tim Sheard and David Stemple. Automatic verification of database transaction safety. *ACM Trans. on Database Systems*, 14(3):322–368, 1989.
12. D. Spelt and H. Balsters. Automatic verification of transactions on an object-oriented database. In *Proc. of the 6th Int. Workshop on Database Programming Languages (DBPL'97)*, volume 1369 of *Lecture Notes in Computer Science*, pages 396–412. Springer, 1998.
13. Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.