Managing Change in Graph-structured Data Using Description Logics (long version with appendix) *

Shqiponja Ahmetaj AHMETAJ@DBAI.TUWIEN.AC.AT
Vienna University of Technology, Austria

Diego Calvanese CALVANESE@INF.UNIBZ.IT
Free University of Bozen-Bolzano, Italy

Magdalena Ortiz ORTIZ@KR.TUWIEN.AC.AT
Vienna University of Technology, Austria

Mantas Šimkus SIMKUS@DBAI.TUWIEN.AC.AT
Vienna University of Technology, Austria

Abstract

In this paper, we consider the setting of graph-structured data that evolves as a result of operations carried out by users or applications. We study different reasoning problems, which range from ensuring the satisfaction of a given set of integrity constraints after a given sequence of updates, to deciding the (non-)existence of a sequence of actions that would take the data to an (un)desirable state, starting either from a specific data instance or from an incomplete description of it. We consider an action language in which actions are finite sequences of conditional insertions and deletions of nodes and labels, and use Description Logics for describing integrity constraints and (partial) states of the data. We then formalize the above data management problems as a static verification problem and several planning problems. We provide algorithms and tight complexity bounds for the formalized problems, both for an expressive DL and for a variant of DL-Lite.

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) [32] has gained popularity recently[1] 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 [14], object-oriented data models, and XML.

In GSD, information is represented by means of a node and edge labeled graph, in which the labels convey semantic information. The representation structures underlying many standard knowledge representation formalisms, and in particular Description Logics (DLs) [5] are paradigmatic examples of GSD. Indeed, in DLs the domain of interest is modeled by means of unary relations (a.k.a. *concepts*) and binary relations (a.k.a. *roles*), and hence the first-order interpretations of a DL knowledge base (KB) can be viewed as node and edge labeled graphs. DLs have been advocated as a proper tool for data management [26], and are very natural for describing complex knowledge about domains represented as GSD. A DL KB comprises an assertional component, called *ABox*, which is often viewed as a possibly incomplete instance of GSD, and a logical theory called terminology or *TBox*, which can be used to infer implicit information from the assertions in the ABox. An alternative possibility is to view the *finite* structures over which DLs are interpreted as (complete) GSD, and the KB as a description of constraints and properties of the data. Taking this view, DLs

$*$. This paper is a extended version of [1] that contains an appendix with proofs.
1. Graph structured data models have their roots in work done in the early '90s, see, e.g., [20].

have been applied, for example, for the static analysis of traditional data models, such as UML class diagrams [11] and Entity Relationship schemata [3]. Problems such as the consistency of a diagram are reduced to KB satisfiability in a suitable DL, and DL reasoning services become tools for managing GSD.

In this paper, we follow the latter view, but aim at using DLs not only for static reasoning about data models, but also for reasoning about the evolution and change over time of GSD that happens as the result of executing actions. The development of automated tools to support such tasks is becoming a pressing problem, given the large amounts and complexity of GSD currently available. Having tools to understand the properties and effects of actions is important and provides added value for many purposes, including application development, integrity preservation, security, and optimization. Questions of interest are, e.g.:

- Will the execution of a given action *preserve* the integrity constraints, for every initial data instance?
- Is there a sequence of actions that leads a given data instance into a state where some property (either desired or not) holds?
- Does a given sequence of actions lead every possible initial data instance into a state where some property necessarily holds?

The first question is analogous to a classic problem in relational databases: verifying *consistency* of database transactions. The second and third questions are classic questions in AI (called *planning* and *projection*, respectively).

In this paper we address these and other related questions, develop tools to answer them, and characterize the computational properties of the underlying problems. The role of DLs in our setting is manifold, and we propose a very expressive DL that is suitable for: *(i)* modeling sophisticated domain knowledge, *(ii)* specifying conditions on the state that should be reached (goal state), and *(iii)* specifying actions to evolve GSD over time. For the latter, we introduce a simple yet powerful language in which actions are finite sequences of (possibly conditional) insertions and deletions performed on concepts and roles, using complex DL concepts and roles as queries. Our results are quite general and allow for analyzing data evolution in several practically relevant settings, including RDF data under constraints expressed in RDFS or OWL. Via the standard reification technique [11], they also apply to the more traditional setting of relational data under schemas expressed in conceptual models (e.g., ER schemas, or UML class diagrams), or to object-oriented data.

In this setting, we address first the *static verification problem*, that is, the problem of verifying whether for every possible state satisfying a given set of constraints (i.e., a given KB), the constraints are still satisfied in the state resulting from the execution of a given (complex) action. We develop a novel technique similar in spirit to *regression* in reasoning about actions [27], and are able to show that static verification is decidable. We provide tight complexity bounds for it, using two different DLs as domain languages. Specifically, we provide a tight coNExpTime bound for the considered expressive DL, and a tight coNP bound for a variation of *DL-Lite* [16]. For our setting, we then study different variants of planning. We define a plan as a sequence of actions that leads a given structure into a state where some property (either desired or not) holds. Then we study problems such as deciding the existence of a plan, both for the case where the initial structure is fully known, and where only a partial description of it is available, and deciding whether a given sequence of actions is always a plan for some goal. Since the existence of a plan (of unbounded length) is undecidable in general, even for lightweight DLs and resctricted actions, we also study plans of bounded length. We provide tight complexity bounds for the different considered variants of the problem, both for lightweight and for expressive DLs. This paper adds an appendix with proofs to [1], some of the results were published in preliminary form [19].

2. An Expressive DL for Modeling GSD

We now define the DL $\mathcal{ALCHOIQ}br$, used to express constraints on GSD. It extends the standard $\mathcal{ALCHOIQ}$ with Boolean combinations of axioms, a constructor for a singleton role, union, difference and restrictions of roles, and variables as place-holders for individuals. The importance of these constructors will be become clear in Sections 3 and 4.

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 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; 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)* 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 α_1, α_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*. We define ($\mathcal{ALCHOIQ}br$-)*formulae* inductively: *(i)* every inclusion and every assertion is a formula; *(ii)* if $\mathcal{K}_1$, $\mathcal{K}_2$ are formulae, so are $\mathcal{K}_1 \wedge \mathcal{K}_2$, $\mathcal{K}_1 \vee \mathcal{K}_2$, and $\dot{\neg}\mathcal{K}_1$. A formula $\mathcal{K}$ with no variables is called *knowledge base (KB)*.

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 $A \in \mathsf{N_C}$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ 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 $\{(o_1, o_2)\}^{\mathcal{I}} = \{(o_1^{\mathcal{I}}, o_2^{\mathcal{I}})\}$, and for ordinary roles of the form $r|_C$, 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, see [5]. 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* β (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.

A NExpTime lower bound for finite satisfiability in $\mathcal{ALCHOIQ}br$ follows from the work of Tobies [35]. Using well-known techniques due to Borgida [13], a matching upper bound can be shown by a direct translation into the two variable fragment with counting, for which finite satisfiability is in NExpTime [31]. Hence, 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.*

We are interested in the problem of *effectively* managing GSD satisfying the knowledge represented 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 $\mathcal{K}$.

3. Updating Graph Structured Data

We now define an action language for manipulating GSD, i.e., finite interpretations. 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 are instances of complex concepts and roles. Since our DL supports nominals $\{o\}$ and singleton roles $\{(o, o')\}$, actions can be defined to add/remove a single individual to/from a concept, or a pair of individuals to/from a role. We allow also for action composition and conditional actions. Note that the action language introduced here is a slight generalization of the one in [19].

Definition 1 (Action language)**.** *A* basic action β *is defined by the following grammar:*

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

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

$$\alpha \longrightarrow \varepsilon \mid \beta \cdot \alpha \mid (\mathcal{K}\,?\,\alpha[\![\alpha]\!]) \cdot \alpha$$

where β is a basic action, $\mathcal{K}$ is an arbitrary $\mathcal{ALCHOIQ}br$-formula, and ε denotes the empty action.

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 $A^{\mathcal{I}}$. Similarly, $(A \ominus C)$ stands for the removal of $C^{\mathcal{I}}$ from $A^{\mathcal{I}}$. The two operations can also be performed on extensions of roles. Composition stands for successive action execution, and a conditional action $\mathcal{K}\,?\,\alpha_1[\![\alpha_2]\!]$ expresses that α_1 is executed if the interpretation is a model of $\mathcal{K}$, and α_2 is executed otherwise. If $\alpha_2 = \varepsilon$ then we have an action with a simple *pre-condition* as in classical planning languages, and we write it as $\mathcal{K}\,?\,\alpha_1$, omitting α_2.

To formally define the semantics of actions, we first introduce the notion of *interpretation update*.

Definition 2 (Interpretation update)**.** *Assume an interpretation $\mathcal{I}$ and let E be a concept or role name. If E is a concept, let $W \subseteq \Delta^{\mathcal{I}}$, otherwise, if E is a role, let $W \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Then, $\mathcal{I} \oplus_E W$ (resp., $\mathcal{I} \ominus_E W$) denotes the interpretation $\mathcal{I}'$ such that $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$, and*
- $E^{\mathcal{I}'} = E^{\mathcal{I}} \cup W$ (resp., $E^{\mathcal{I}'} = E^{\mathcal{I}} \setminus W$), and
- $E_1^{\mathcal{I}'} = E_1^{\mathcal{I}}$, for all symbols $E_1 \neq E$.

Now we can define the semantics of ground actions:

Definition 3. *Given a ground action α, we define a mapping S_α from interpretations to interpretations as follows:*

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

In the following, we assume that interpretations are updated using the above language.

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 three employees that work in the active projects.*

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

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 following action α_1 captures the termination of project p_1, which is removed from the active projects and added to the finished ones. The employees working only for this project are removed.

$$\begin{array}{rl}
\alpha_1 = & \mathsf{ActivePrj} \ominus \{\mathsf{p_1}\} \cdot \mathsf{FinishedPrj} \oplus \{\mathsf{p_1}\} \cdot \\
& \mathsf{Empl} \ominus \forall \mathsf{worksFor}.\{\mathsf{p_1}\}
\end{array}$$

The interpretation $S_{\alpha_1}(\mathcal{I}_1)$ that reflects the status of the database after action α_1 looks as follows:

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

Note that we have not defined the semantics of actions with variables, i.e., for non-ground actions. 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.

Example 2. *The following action α_2 with variables x, y, z transfers the employee x from project y to project z:*

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

Under the substitution σ with $\sigma(x) = \mathsf{e}_1$, $\sigma(y) = \mathsf{p}_1$, and $\sigma(z) = \mathsf{p}_2$, the action α_2 first checks whether e_1 is an (instance of) employee, p_1, p_2 are projects, and e_1 works for p_1. If yes, it removes the worksFor *link between e_1 and p_1, and creates a* worksFor *link between e_1 and p_2. If any of the checks fails, it does nothing.*

4. Capturing Action Effects

In this section we present our core technical tool: a transformation $\mathsf{TR}_\alpha(\mathcal{K})$ that rewrites $\mathcal{K}$ incorporating the possible effects of an action α. Intuitively, the models of $\mathsf{TR}_\alpha(\mathcal{K})$ are exactly the interpretations $\mathcal{I}$ such that applying α on $\mathcal{I}$ leads to a model of $\mathcal{K}$. In this way, we can effectively reduce reasoning about changes in any database that satisfies a given $\mathcal{K}$, to reasoning about a single KB. In the next section we use this transformation to solve a wide range of data management problems by reducing them to standard DL reasoning services, such as finite (un)satisfiability. This transformation can be seen as a form of *regression* [27], which incorporates the effects of a sequence of actions 'backwards', from the last one to the first one.

Definition 4. *Given a KB $\mathcal{K}$, we use $\mathcal{K}_{E\leftarrow E'}$ to denote the KB that is obtained from $\mathcal{K}$ by replacing every name E by the (possibly more complex) expression E'. Given a KB $\mathcal{K}$ and an action α, we define $\mathsf{TR}_\alpha(\mathcal{K})$ as follows:*

$$\begin{aligned}\mathsf{TR}_\varepsilon(\mathcal{K}) &= \mathcal{K}\\ \mathsf{TR}_{(A\oplus C)\cdot\alpha}(\mathcal{K}) &= (\mathsf{TR}_\alpha(\mathcal{K}))_{A\leftarrow A\sqcup C}\\ \mathsf{TR}_{(A\ominus C)\cdot\alpha}(\mathcal{K}) &= (\mathsf{TR}_\alpha(\mathcal{K}))_{A\leftarrow A\sqcap\neg C}\\ \mathsf{TR}_{(p\oplus r)\cdot\alpha}(\mathcal{K}) &= (\mathsf{TR}_\alpha(\mathcal{K}))_{p\leftarrow p\cup r}\\ \mathsf{TR}_{(p\ominus r)\cdot\alpha}(\mathcal{K}) &= (\mathsf{TR}_\alpha(\mathcal{K}))_{p\leftarrow p\setminus r}\\ \mathsf{TR}_{(\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!])\cdot\alpha}(\mathcal{K}) &= (\dot\neg\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1\cdot\alpha}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2\cdot\alpha}(\mathcal{K})).\end{aligned}$$

Note that the size of $\mathsf{TR}_\alpha(\mathcal{K})$ might be exponential in the size of α. We now show that this transformation correctly captures the effects of complex actions.

Theorem 2. *Assume a ground action α and a KB $\mathcal{K}$. For every interpretation $\mathcal{I}$, we have $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$.*

Proof. We define $s(\alpha)$ as follows: $s(\varepsilon) = 0$, $s(\beta \cdot \alpha) = 1 + s(\alpha)$, and $s(\mathcal{K}\,?\,\alpha_1[\![\alpha_2]\!] \cdot \alpha_3) = 1 + s(\alpha_1) + s(\alpha_2) + s(\alpha_3)$. We prove the claim by induction on $s(\alpha)$. In the base case where $s(\alpha) = 0$ and $\alpha = \varepsilon$, 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 $\mathcal{I}' = \mathcal{I} \oplus_A C^\mathcal{I}$, that is, $\mathcal{I}'$ coincides with $\mathcal{I}$ except that $A^{\mathcal{I}'} = A^\mathcal{I} \cup C^\mathcal{I}$. For every KB $\mathcal{K}'$, $\mathcal{I}' \models \mathcal{K}'$ iff $\mathcal{I} \models \mathcal{K}'_{A\leftarrow A\sqcup C}$ (This can be proved by a straightforward induction on the structure of the expressions in $\mathcal{K}'$). 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 $S_{\alpha'}(\mathcal{I}') = S_{\alpha'}(S_{(A\oplus C)}(\mathcal{I})) = S_\alpha(\mathcal{I})$ by definition, we obtain $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$ iff $S_\alpha(\mathcal{I}) \models \mathcal{K}$ as desired.

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

Finally, we consider $\alpha = (\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!]) \cdot \alpha'$, and assume an arbitrary $\mathcal{I}$. We consider the case where $\mathcal{I} \models \mathcal{K}_1$; the case where $\mathcal{I} \not\models \mathcal{K}_1$ is analogous. By definition $S_\alpha(\mathcal{I}) = S_{\alpha_1\cdot\alpha'}(\mathcal{I})$. By the induction hypothesis we know that $S_{\alpha_1\cdot\alpha'}(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdot\alpha'}(\mathcal{K})$, so $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdot\alpha'}(\mathcal{K})$. Since $\mathcal{I} \models \mathcal{K}_1$ and $\mathsf{TR}_{(\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!])\cdot\alpha}(\mathcal{K}) = (\dot\neg\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1\cdot\alpha}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2\cdot\alpha}(\mathcal{K}))$, it follows that $S_\alpha(\mathcal{I}) \models \mathcal{K}$ iff $\mathcal{I} \models \mathsf{TR}_{(\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!])\cdot\alpha}(\mathcal{K})$. □

This theorem will be important for solving the reasoning problems we study below.

Example 3. *The following KB $\mathcal{K}_1$ expresses constraints on the project database of our running example: all projects are active or finished, the domain of* worksFor *are the employees, and its range the projects.*

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

By applying the transformation above to $\mathcal{K}_1$ and α_1, we obtain the following KB $\mathsf{TR}_{\alpha_1}(\mathcal{K}_1)$:

$$\begin{aligned}&(\mathsf{Prj} \sqsubseteq (\mathsf{ActivePrj} \sqcap \neg\{\mathsf{p}_1\}) \sqcup (\mathsf{FinishedPrj} \sqcup \{\mathsf{p}_1\})) \wedge \\ &(\exists \mathsf{worksFor}.\top \sqsubseteq \mathsf{Empl} \sqcap \exists \mathsf{worksFor}.\neg\{\mathsf{p}_1\}) \wedge \\ &(\exists \mathsf{worksFor}^-.\top \sqsubseteq \mathsf{Prj})\end{aligned}$$

5. Static Verification

In this section, we consider the scenario where DL KBs are used to impose integrity constraints on GSD. One of the most basic reasoning problems for action analysis in this setting is *static verification*, which consists in checking whether the execution of an action α always preserves the satisfaction of integrity constraints given by a KB.

Definition 5 (The static verification problem)**.** *Let $\mathcal{K}$ be a KB. We say that an action α is $\mathcal{K}$-*preserving *if for every ground instance α' of α 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 defined as follows:*

(SV) Given an action α and a KB $\mathcal{K}$, is α $\mathcal{K}$-preserving?

Using the transformation $\mathsf{TR}_\alpha(\mathcal{K})$ above, we can reduce static verification to finite (un)satisfiability of $\mathcal{ALCHOIQ}br$ KBs: An action α is not $\mathcal{K}$-preserving iff some finite model of $\mathcal{K}$ does not satisfy $\mathsf{TR}_{\alpha^*}(\mathcal{K})$, where α^* is a 'canonical' grounding of α. Formally, we have:

Theorem 3. *Assume a (complex) action α and a KB $\mathcal{K}$. Then the following are equivalent:*

(i) The action α is not $\mathcal{K}$-preserving.

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

Example 4. *The action α_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 $\exists\mathsf{worksFor}.\mathsf{Prj} \sqsubseteq \mathsf{Empl}$ is violated. This is reflected in the fact that $\mathcal{I}_1 \not\models \mathsf{TR}_{\alpha_1}(\mathcal{K}_1)$, as can be readily checked. Intuitively, values removed from* Empl *should also be removed from* worksFor*, as in the following $\mathcal{K}_1$-preserving action:*

$$\begin{aligned}\alpha_1' = \;&\mathsf{ActivePrj} \ominus \{\mathsf{p}_1\} \cdot \mathsf{FinishedPrj} \oplus \{\mathsf{p}_1\} \cdot \\ &\mathsf{Empl} \ominus \forall\mathsf{worksFor}.\{\mathsf{p}_1\} \cdot \mathsf{worksFor} \ominus \mathsf{worksFor}|_{\{\mathsf{p}_1\}}\end{aligned}$$

The above theorem provides an algorithm for static verification, which we can also use to obtain tight bounds on the computational complexity of the problem. Indeed, even though $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ may be of size exponential in α, we can avoid to generate it all at once. More precisely, we use a non-deterministic polynomial time many-one reduction that builds only $\mathcal{K} \wedge \dot\neg\mathsf{TR}^c_{\alpha^*}(\mathcal{K})$ for a fragment $\dot\neg\mathsf{TR}^c_\alpha(\mathcal{K})$ of $\dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ that corresponds to one fixed way of choosing one of α_1 or α_2 for each conditional action $\mathcal{K}' \,?\, \alpha_1 [\![\alpha_2]\!]$ in α (intuitively, we can view $\dot\neg\mathsf{TR}^c_{\alpha^*}(\mathcal{K})$ as one conjunct of the DNF of $\dot\neg\mathsf{TR}_\alpha(\mathcal{K})$, where axioms and assertions are treated as propositions). Such a $\dot\neg\mathsf{TR}^c_\alpha(\mathcal{K})$ has polynomial size, and it can be built non-deterministically in polynomial time. It is not hard to show that $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable iff there is some choice $\mathsf{TR}^c_{\alpha^*}(\mathcal{K})$ such that $\mathcal{K} \wedge \dot\neg\mathsf{TR}^c_{\alpha^*}(\mathcal{K})$ is finitely satisfiable. By Theorem 1, the latter test can be done in non-deterministic exponential time, hence from Theorem 3 we obtain:

Theorem 4. *The problem (SV) is co*NEXPTIME*-complete in case the input KB is expressed in $\mathcal{ALCHOIQ}br$.*

We note that in our definition of the (SV) problem, in addition to the action to be verified, one has as input only one KB $\mathcal{K}$ expressing constraints. We can also consider other interesting variations of the problem where, for example, we have a pair of KBs $\mathcal{K}_{pre}$ and $\mathcal{K}_{post}$ instead of (or in addition to) $\mathcal{K}$ and we want to decide whether executing the action on any model of $\mathcal{K}_{pre}$ (and $\mathcal{K}$) leads to a model of $\mathcal{K}_{post}$ (and $\mathcal{K}$). The reasoning techniques and upper bounds presented above also apply to these generalized settings.

Lowering the Complexity

The goal of this section is to identify a setting for which the computational complexity of static verification is lower. The natural way to achieve this is to consider as constraint language a DL with better computational properties, such as the logics of the *DL-Lite* family [16].

Unfortunately, we cannot achieve tractability, since static verification is coNP hard even in a very restricted setting, as shown next.

Theorem 5. *The static verification problem is co*NP*-hard already for KBs of the form* $(A_1 \sqsubseteq \neg A'_1) \wedge \cdots \wedge (A_n \sqsubseteq \neg A'_n)$*, where each* A_i, A'_i *is a concept name, and ground sequences of basic actions of the forms* $(A \oplus C)$ *and* $(A \ominus C)$.

We next present a rich variant of *DL-Lite*$_{\mathcal{R}}$, which we call *DL-Lite*$^{+}_{\mathcal{R}}$, for which the static verification problem is in coNP. It supports (restricted) Boolean combinations of inclusions and assertions, and allows for complex concepts and roles in assertions. As shown below, this allows us to express the effects of actions inside *DL-Lite*$^{+}_{\mathcal{R}}$ KBs.

Definition 6. *The logic DL-Lite*$^{+}_{\mathcal{R}}$ *is defined as follows:*

- *Concept inclusions have the form* $C_1 \sqsubseteq C_2$ *or* $C_1 \sqsubseteq \neg C_2$*, with* $C_1, C_2 \in \mathsf{N_C} \cup \{\exists p.\top, \exists p^-.\top \mid p \in \mathsf{N_R}\}$.
- *Role inclusions in* $\mathcal{K}$ *have the form* $r_1 \sqsubseteq r_2$ *or* $r_1 \sqsubseteq \neg r_2$*, with* $r_1, r_2 \in \mathsf{N_R} \cup \{p^- \mid p \in \mathsf{N_R}\}$.
- *Role assertions are defined as for* $\mathcal{ALCHOIQ}br$*, but in concept assertions* $o : C$*, we require* $C \in \mathbf{B}^+$*, 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* $B_1, B_2 \in \mathbf{B}^+$.
- *Formulae and KBs are defined as for* $\mathcal{ALCHOIQ}br$*, but the operator* $\dot\neg$ *may occur only in front of assertions.*

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, and*
- *all assertions in* $\mathcal{K}$ *are* basic assertions *of the forms* $o : A$ *with* $A \in \mathsf{N_C}$*, and* $(o, o') : p$ *with* $p \in \mathsf{N_R}$.

We make the unique name assumption (UNA)*: for every pair of individuals* o_1*,* o_2 *and interpretation* $\mathcal{I}$*, we have* $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$.

We need to slightly restrict the action language, which involves allowing only Boolean combinations of assertions to express the condition $\mathcal{K}$ in actions of the form $\mathcal{K}\,?\,\alpha_1[\![\alpha_2]\!]$.

Definition 7. *A (complex) action* α *is called* simple *if (i) no (concept or role) inclusions occur in* α*, and (ii) all concepts of* α *are from* $\mathbf{B}^+$.

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

Theorem 6. *Finite satisfiability of DL-Lite*$^{+}_{\mathcal{R}}$ *KBs is* NP*-complete.*

DL-Lite$_{\mathcal{R}}^{+}$ is expressive enough to allow us to reduce static verification for simple actions to finite unsatisfiability, and similarly as above, we can use a non-deterministic polynomial time many-one reduction (from the complement of static verification to finite unsatisfiability) to obtain a coNP upper bound on the complexity of static verification. This bound is tight, even if we allow only actions with preconditions rather than full conditional actions. We note that all lower bounds in the next section also hold for this restricted case.

Theorem 7. *The static verification problem for DL-Lite*$_{\mathcal{R}}^{+}$ *KBs and simple actions is co*NP*-complete.*

6. Planning

We have focused so far on ensuring that the satisfaction of constraints is preserved when we evolve GSD. But additionally, there may be desirable states of the GSD that we want to achieve, or undesirable ones that we want to avoid. For instance, one may want to ensure that a finished project is never made active again. This raises several problems, such as deciding if there exists a sequence of actions to reach a state with certain properties, or whether a given sequence of actions always ensures that a state with certain properties is reached. We consider now these problems and formalize them by means of *automated planning*.

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 an agent from the current state to a state that satisfies a given goal.

Definition 8. *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* $\langle \alpha_1, \ldots, \alpha_n \rangle$ *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* $\mathcal{I}' = \langle \Delta^{\mathcal{I}} \cup \Delta, \cdot^{\mathcal{I}} \rangle$.

Recall that actions in our setting do not modify the domain of an interpretation. To support unbounded introduction of values in the data, the definition of planning above allows for the domain to be expanded a-priori with a finite set of fresh domain elements.

We can now define the first planning problems we study:

(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.

Example 5. *Recall the interpretation* $\mathcal{I}_1$ *and the action* α_1' *from Example 4, and the substitution* σ *from Example 2, which gives us the following ground instance of* α_2*:*

$$\begin{aligned} \alpha_2' = {} & (\mathsf{e}_1 : \mathsf{Empl} \wedge \mathsf{p}_1 : \mathsf{Prj} \wedge \mathsf{p}_2 : \mathsf{Prj} \wedge (\mathsf{e}_1, \mathsf{p}_1) : \mathsf{worksFor})\,? \\ & (\mathsf{worksFor} \ominus \{(\mathsf{e}_1, \mathsf{p}_1)\} \cdot \mathsf{worksFor} \oplus \{(\mathsf{e}_1, \mathsf{p}_2)\}) \end{aligned}$$

The following goal *KB requires that* p_1 *is not an active project, and that* e_1 *is an employee.*

$$\mathcal{K}_g = \dot{\neg}(\mathsf{p}_1 : \mathsf{ActivePrj}) \wedge \mathsf{e}_1 : \mathsf{Empl}$$

A plan for $\mathcal{K}_g$ from $\mathcal{I}_1$ is the sequence of actions $\langle \alpha_2', \alpha_1' \rangle$. The interpretation $S_{\alpha_2' \cdot \alpha_1'}(\mathcal{I}_1)$ that reflects the status of the data after applying $\langle \alpha_2', \alpha_1' \rangle$ looks as follows:

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

Clearly, $S_{\alpha_2' \cdot \alpha_1'}(\mathcal{I}_1) \models \mathcal{K}_1$.

Unfortunately, these problems are undecidable in general, which can be shown by a reduction from the Halting problem for Turing machines.

Theorem 8. *The problems (P1) and (P2) are undecidable, already for DL-Lite$^+_\mathcal{R}$ KBs and simple actions.*

Intuitively, problem (P1) is undecidable because we cannot know how many fresh objects need to be added to the domain of $\mathcal{I}$, but it becomes decidable if the size of Δ in Definition 8 is bounded. It is not difficult to see that problem (P2) remains undecidable even if the domain is assumed fixed (as the problem definition quantifies existentially over interpretations, one can choose interpretations with sufficiently large domains). However, also (P2) becomes decidable if we place a bound on the length of plans. More precisely, the following problems are decidable.

(P1$_\mathrm{b}$) 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 $|\Delta| \leq k$?

(P2$_\mathrm{b}$) 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 $\leq k$ for $\sigma(\mathcal{K})$ from some finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$?

We now study the complexity of these problems, assuming that the input bounds k are coded in unary. The problem (P1$_\mathrm{b}$) can be solved in polynomial space, and thus is not harder than deciding the existence of a plan in standard automated planning formalisms such as propositional STRIPS [15]. In fact, the following lower bound can be proved by a reduction from the latter formalism, or by an adaptation of the Turing Machine reduction used to prove undecidability in Theorem 8.

Theorem 9. *The problem (P1$_b$) is* PSPACE*-complete for $\mathcal{ALCHOIQ}br$ KBs.*

Now we establish the complexity of (P2$_\mathrm{b}$), both in the general setting (i.e., when $\mathcal{K}_{pre}$ and $\mathcal{K}$ are in $\mathcal{ALCHOIQ}br$), and for the restricted case of *DL-Lite*$^+_\mathcal{R}$ KBs and simple actions. For (SV), considering the latter setting allowed us to reduce the complexity from coNEXPTIME to coNP. Here we obtain an analogous result and go from NEXPTIME-completeness to NP-completeness.

Theorem 10. *The problem (P2$_b$) is* NEXPTIME*-complete. It is* NP*-complete if $\mathcal{K}_{pre}, \mathcal{K}$ are expressed in DL-Lite$^+_\mathcal{R}$ and all actions in Act are simple.*

Now we consider three problems that are related to ensuring plans that *always* achieve a given goal, no matter what the initial data is. They are variants of the so-called *conformant* planning, which deals with planning under various forms of incomplete information. In our case, we assume that we have an incomplete description of the initial state, since we only know it satisfies a given precondition, but have no concrete interpretation.

The first of such problems is to 'certify' that a candidate plan is indeed a plan for the goal, for every possible database satisfying the precondition.

(C) Given a sequence $P = \langle \alpha_1, \ldots, \alpha_n \rangle$ 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 possible substitution σ?

Finally, we are interested in 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. We formulate the problem with and without a bound on the length of the plans we are looking for.

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

(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 such that $\sigma(P)$ is of length at most k and is a plan for $\sigma(\mathcal{K})$ from every finite interpretation $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, for every possible substitution σ?

We conclude with the complexity of these problems:

Theorem 11. *The following hold:*

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

7. Related Work

Using DLs to understand the properties of systems while fully taking into account both structural and dynamic aspects is very challenging [36]. Reasoning in DLs extended with a temporal dimension becomes quickly undecidable [2], unless severe restrictions on the expressive power of the DL are imposed [4]. An alternative approach to achieve decidability is to take a so-called "functional view of KBs" [28], according to which each state of the KB can be queried via logical implication, and the KB is progressed from one state to the next through forms of update [17]. This makes it possible (under suitable conditions) to *statically verify* (temporal) integrity constraints over the evolution of a system [6, 10].

Updating databases, and logic theories in general, is a classic topic in knowledge representation, discussed extensively in the literature, cf. [21, 23]. The updates described by our action language are similar in spirit to the knowledge base updates studied in other works, and in particular, the ABox updates considered in [29], and [24]. As our updates are done directly on interpretations rather than on (the instance level of) knowledge bases, we do not encounter the expressibility and succinctness problems faced there.

Concerning the reasoning problems we tackle, verifying consistency of transactions is a crucial problem that has been studied extensively in Databases. It has been considered for different kinds of transactions and constraints, over traditional relational databases [33], object-oriented databases [34, 12], and deductive databases [25], to name a few. Most of these works adopt expressive formalisms like (extensions of) first or higher order predicate logic [12], or undecidable tailored languages [33] to express the constraints and the operations on the data. Verification systems are often implemented using theorem provers, and complete algorithms cannot be devised.

As mentioned, the problems studied in Section 6 are closely related to automated planning, a topic extensively studied in AI. DLs have been employed to reason about actions, goals, and plans, as well as about the application domains in which planning is deployed, see [22] and its references. Most relevant to us is the significant body of work on DL-based action languages [8, 30, 7, 29, 9]. In these formalisms, DL constructs are used to give conditions on the effects of action execution, which are often non-deterministic. A central problem considered is the *projection problem*, which consists in deciding whether every possible execution of an action sequence on a possibly incomplete state will lead to a state that satisfies a given property. Clearly, our certification problem (C), which involves an incomplete initial state, is a variation of the projection problem. However, we do not face the challenge of having to consider different possible executions of non-deterministic actions. Many of our other reasoning problems are similar to problems considered in these works, in different forms and contexts. A crucial difference is that our well-behaved action language allows us to obtain decidability even when we employ full-fledged TBoxes for specifying goals, preconditions, and domain constraints. To the best of our knowledge, previous results rely on TBox acyclicity to ensure decidability.

8. Conclusions

We have considered graph structured data that evolve as a result of updates expressed in a powerful yet well-behaved action language. We have studied several reasoning problems that support the static analysis of actions and their effects on the state of the data. We have shown the decidability of most problems, and in the cases where the general problem is undecidable, we have identified decidable restrictions and have characterized the computational complexity for a very expressive DL and a variant of *DL-Lite*. We believe this work provides powerful tools for analyzing the effects of executing complex actions on databases, possibly in the presence of integrity constraints expressed in rich DLs. Our upper bounds rely on a novel KB transformation technique, which enables to reduce most of the reasoning tasks to finite (un)satisfiability in a DL. This calls for developing finite model reasoners for DLs (we note that $\mathcal{ALCHOIQ}br$ does not have the finite model property). It also remains to better understand the complexity of finite model reasoning in different variations of *DL-Lite*. E.g., extensions of *DL-Lite*$^{+}_{\mathcal{R}}$ with role functionality would be very useful in the context of graph structured data. Generalizing the positive decidability results to logics with powerful identification constraints, like the ones considered in [18], would also be of practical importance. Given that the considered problems are intractable even for weak fragments of the core *DL-Lite* and very restricted forms of actions, it remains to explore how feasible these tasks are in practice, and whether there are meaningful restrictions that make them tractable.

Acknowledgments

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.

References

[1] Shqiponja Ahmetaj, Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Managing change in Graph-structured Data using Description Logics. In *Proc. of AAAI 2014*, 2014.

[2] Alessandro Artale. Reasoning on temporal class diagrams: Undecidability results. *AMAI*, 46(3):265–288, 2006.

[3] Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Reasoning over extended ER models. In *Proc. of ER*, volume 4801 of *LNCS*, pages 277–292. Springer, 2007.

[4] Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Tailoring temporal description logics for reasoning over temporal conceptual models. In *Proc. of FroCoS*, pages 1–11. Springer, 2011.

[5] 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.

[6] Franz Baader, Silvio Ghilardi, and Carsten Lutz. LTL over description logic axioms. *ACM TOCL*, 13(3):21:1–21:32, 2012.

[7] Franz Baader, Marcel Lippmann, and Hongkai Liu. Using causal relationships to deal with the ramification problem in action formalisms based on description logics. In *Proc. of LPAR 17*, volume 6397 of *LNCS*, pages 82–96. Springer, 2010.

[8] Franz Baader, Carsten Lutz, Maja Milicic, Ulrike Sattler, and Frank Wolter. Integrating description logics and action formalisms: First results. In *Proc. of AAAI*, pages 572–577, 2005.

[9] Franz Baader and Benjamin Zarrie. Verification of Golog programs over description logic actions. In *Proc. of FroCoS*, volume 8152 of *LNCS*, pages 181–196. Springer, 2013.

[10] Babak Bagheri Hariri, Diego Calvanese, Marco Montali, Giuseppe De Giacomo, Riccardo De Masellis, and Paolo Felli. Description logic Knowledge and Action Bases. *JAIR*, 46:651–686, 2013.

[11] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *AIJ*, 168(1–2):70–118, 2005.

[12] Anthony J. Bonner and Michael Kifer. An overview of Transaction Logic. *TCS*, 133(2):205–265, 1994.

[13] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *AIJ*, 82(1–2):353–367, 1996.

[14] Dan Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, W3C, February 2004. Available at http://www.w3.org/TR/rdf-schema/.

[15] Tom Bylander. The computational complexity of propositional STRIPS planning. *AIJ*, 69:165–204, 1994.

[16] 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. *JAR*, 39(3):385–429, 2007.

[17] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Actions and programs over description logic knowledge bases: A functional approach. In *Knowing, Reasoning, and Acting: Essays in Honour of Hector Levesque*. College Publications, 2011.

[18] Diego Calvanese, Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, and Mantas Šimkus. Capturing relational schemas and functional dependencies in rdfs. In *Proc. of AAAI 2014*, 2014.

[19] Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Evolving graph databases under description logic constraints. In *Proc. of DL*, volume 1014 of *CEUR Workshop Proceedings*, pages 120–131, 2013.

[20] Mariano P. Consens and Alberto O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *Proc. of PODS*, pages 404–416, 1990.

[21] Ronald Fagin, Gabriel M. Kuper, Jeffrey D. Ullman, and Moshe Y. Vardi. Updating logical databases. In *Advances in Computing Research*, pages 1–18. JAI Press, 1986.

[22] Yolanda Gil. Description logics and planning. *AI Magazine*, 26(2):73–84, 2005.

[23] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. of KR*, pages 387–394, 1991.

[24] Evgeny Kharlamov, Dmitriy Zheleznyakov, and Diego Calvanese. Capturing model-based ontology evolution at the instance level: The case of *DL-Lite*. *JCSS*, 79(6):835–872, 2013.

[25] Robert A. Kowalski, Fariba Sadri, and Paul Soper. Integrity checking in deductive databases. In *Proc. of VLDB*, pages 61–69, 1987.

[26] Maurizio Lenzerini. Ontology-based data management. In *Proc. of CIKM*, pages 5–6, 2011.

[27] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *JLP*, 31:59–84, 1997.

[28] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *AIJ*, 23:155–212, 1984.

[29] Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Foundations of instance level updates in expressive description logics. *AIJ*, 175(18):2170–2197, 2011.

[30] Maja Milicic. *Action, Time and Space in Description Logics*. PhD thesis, TU Dresden, 2008.

[31] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *JLLI*, 14(3):369–395, 2005.

[32] Sherif Sakr and Eric Pardede, editors. *Graph Data Management: Techniques and Applications*. IGI Global, 2011.

[33] Tim Sheard and David Stemple. Automatic verification of database transaction safety. *ACM TODS*, 14(3):322–368, 1989.

[34] D. Spelt and H. Balsters. Automatic verification of transactions on an object-oriented database. In *Proc. of DBPL*, volume 1369 of *LNCS*, pages 396–412. Springer, 1998.

[35] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *JAIR*, 12:199–217, 2000.

[36] Frank Wolter and Michael Zakharyaschev. Temporalizing description logic. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems*, pages 379–402. Studies Press/Wiley, 1999.

Appendix

Proof of Theorem 3. (i) to *(ii)*. Assume there exist a ground instance α' of α and a finite interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $S_{\alpha'}(\mathcal{I}) \not\models \mathcal{K}$. Then by Theorem 2, $\mathcal{I} \not\models \mathsf{TR}_{\alpha'}(\mathcal{K})$. 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 α into α'. Suppose also $o'_1 \to x_1, \ldots, o'_n \to x_n$ is the substitution that transforms α into α^*. 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 Theorem 2, $S_{\alpha^*}(\mathcal{I}) \not\models \mathcal{K}$. □

Proof of Theorem 4. For coNExpTime-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.

Obtaining a matching upper bound is slightly more involved. It follows from Theorem 3 that the complement of static verification in the presence of $\mathcal{ALCHOIQ}br$ KBs reduces to finite satisfiability of a KB $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ in $\mathcal{ALCHOIQ}br$, but unfortunately, this reduction is exponential in general. Hence we use an alternative reduction that allow us to *non-deterministically* build in polynomial time a formula $\mathcal{K}'$ of polynomial size, such that $\mathcal{K} \wedge \mathcal{K}'$ is satisfiable iff $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is satisfiable. The upper bound then follows from this and the fact that finite satisfiability in $\mathcal{ALCHOIQ}br$ is NExpTime-complete (c.f. Theorem 1).

To obtain this non-deterministic polynomial time many-one reduction, it is convenient to first define a minor variation $\overline{\mathsf{TR}}_\alpha(\mathcal{K})$ of the transformation above, which generates an already negated KB.

$$
\begin{aligned}
\overline{\mathsf{TR}}_\varepsilon(\mathcal{K}) &= \dot\neg\mathcal{K}\\
\overline{\mathsf{TR}}_{(A\oplus C)\cdot\alpha}(\mathcal{K}) &= (\overline{\mathsf{TR}}_\alpha(\mathcal{K}))_{A\leftarrow A\sqcup C}\\
\overline{\mathsf{TR}}_{(A\ominus C)\cdot\alpha}(\mathcal{K}) &= (\overline{\mathsf{TR}}_\alpha(\mathcal{K}))_{A\leftarrow A\sqcap\neg C}\\
\overline{\mathsf{TR}}_{(p\oplus r)\cdot\alpha}(\mathcal{K}) &= (\overline{\mathsf{TR}}_\alpha(\mathcal{K}))_{p\leftarrow p\cup r}\\
\overline{\mathsf{TR}}_{(p\ominus r)\cdot\alpha}(\mathcal{K}) &= (\overline{\mathsf{TR}}_\alpha(\mathcal{K}))_{p\leftarrow p\setminus r}\\
\overline{\mathsf{TR}}_{(\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!])\cdot\alpha}(\mathcal{K}) &= \big(\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1\cdot\alpha}(\mathcal{K})\big) \vee \big(\dot\neg\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_2\cdot\alpha}(\mathcal{K})\big)
\end{aligned}
$$

It can be shown by a straightforward induction on $s(\alpha)$ (as defined in the Proof of Theorem 2) that $\overline{\mathsf{TR}}_\alpha(\mathcal{K})$ is logically equivalent to $\dot\neg\mathsf{TR}_\alpha(\mathcal{K})$ for every $\mathcal{K}$ and every α. Hence, by Theorem 2, $\mathcal{K} \wedge \overline{\mathsf{TR}}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable iff $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable iff α is not $\mathcal{K}$-preserving.

Now, for the desired reduction, we use a non-deterministic version of $\overline{\mathsf{TR}}_\alpha(\mathcal{K})$ that is defined analogously but in the last case, for the conditional axioms, we non-deterministically choose between $\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1\cdot\alpha}(\mathcal{K})$, or $\dot\neg\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_2\cdot\alpha}(\mathcal{K})$, rather than considering the disjunction of both. We denote by $\overline{\mathbf{TR}}_\alpha(\mathcal{K})$ the set of all the KBs obtained this way, that is:

$$
\begin{aligned}
\overline{\mathbf{TR}}_\varepsilon(\mathcal{K}) &= \{\dot\neg\mathcal{K}\}\\
\overline{\mathbf{TR}}_{(A\oplus C)\cdot\alpha}(\mathcal{K}) &= \{\mathcal{K}'_{A\leftarrow A\sqcup C} \mid \mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})\}\\
\overline{\mathbf{TR}}_{(A\ominus C)\cdot\alpha}(\mathcal{K}) &= \{\mathcal{K}'_{A\leftarrow A\sqcap\neg C} \mid \mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})\}\\
\overline{\mathbf{TR}}_{(p\oplus r)\cdot\alpha}(\mathcal{K}) &= \{\mathcal{K}'_{p\leftarrow p\cup r} \mid \mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})\}\\
\overline{\mathbf{TR}}_{(p\ominus r)\cdot\alpha}(\mathcal{K}) &= \{\mathcal{K}'_{p\leftarrow p\setminus r} \mid \mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})\}\\
\overline{\mathbf{TR}}_{(\mathcal{K}_1\,?\,\alpha_1[\![\alpha_2]\!])\cdot\alpha}(\mathcal{K}) &= \{\mathcal{K}_1 \wedge \mathcal{K}' \mid \mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha_1\cdot\alpha}(\mathcal{K})\} \cup \{\dot\neg\mathcal{K}_1 \wedge \mathcal{K}' \mid \mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha_2\cdot\alpha}(\mathcal{K})\}
\end{aligned}
$$

It is easy to see that $|\overline{\mathbf{TR}}_\alpha(\mathcal{K})|$ may be exponential in α and $\mathcal{K}$, but each $\mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ is of polynomial size and can be built (non-deterministically) in polynomial time. It is only left to show that $\mathcal{K} \wedge \overline{\mathsf{TR}}_\alpha(\mathcal{K})$ is finitely satisfiable iff there is some $\mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ such that $\mathcal{K} \wedge \mathcal{K}'$ is finitely satisfiable. This is a consequence of the fact that, for every interpretation $\mathcal{I}$, $\mathcal{I} \models \overline{\mathsf{TR}}_\alpha(\mathcal{K})$ iff there is some $\mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'$.

We show this by induction on $s(\alpha)$. The base case is straightforward: if $\alpha = \epsilon$, then $\overline{\mathbf{TR}}_\alpha(\mathcal{K}) = \{\overline{\mathsf{TR}}_\alpha(\mathcal{K})\}$. For the inductive step, we first consider $\alpha = (A \oplus C) \cdot \alpha'$. First we assume that $\mathcal{I} \models \overline{\mathsf{TR}}_\alpha(\mathcal{K})$. That is, $\mathcal{I} \models (\overline{\mathsf{TR}}_{\alpha'}(\mathcal{K}))_{A \leftarrow A \sqcup C}$. We can apply the induction hypothesis to infer that there exists $\mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha'}(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'_{A \leftarrow A \sqcup C}$, which implies that exists $\mathcal{K}'' = \mathcal{K}'_{A \leftarrow A \sqcup C}$ such that $\mathcal{K}'' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ and $\mathcal{I} \models \mathcal{K}''$ as desired. For the converse, if $\mathcal{I} \models \mathcal{K}''$ for some $\mathcal{K}'' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$, by definition we have that there is some $\mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha'}(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'_{A \leftarrow A \sqcup C}$. Using the induction hypothesis we get $\mathcal{I} \models \overline{\mathbf{TR}}_{\alpha'}(\mathcal{K})_{A \leftarrow A \sqcup C}$, that is, $\mathcal{I} \models \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ as desired. The cases of $\alpha = (A \ominus C) \cdot \alpha'$, $\alpha = (p \oplus r) \cdot \alpha'$, and $\alpha = (p \ominus r) \cdot \alpha'$ are analogous.

Finally, consider $\alpha = (\mathcal{K}_1 \,?\, \alpha_1 [\![\alpha_2]\!]) \cdot \alpha'$. We first show that if $\mathcal{I} \models \overline{\mathsf{TR}}_\alpha(\mathcal{K})$, then there is some $\mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'$. By definition, $\overline{\mathsf{TR}}_\alpha(\mathcal{K}) = \big(\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1 \cdot \alpha}(\mathcal{K})\big) \vee \big(\dot{\neg}\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_2 \cdot \alpha}(\mathcal{K})\big)$. So, if $\mathcal{I} \models \overline{\mathsf{TR}}_\alpha(\mathcal{K})$, then one of $\mathcal{I} \models \mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1 \cdot \alpha}(\mathcal{K})$ or $\mathcal{I} \models \dot{\neg}\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_2 \cdot \alpha}(\mathcal{K})$ holds. In the former case, we can use the induction hypothesis to conclude that there exists some $\mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha_2 \cdot \alpha}(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}_1 \wedge \mathcal{K}'$. Since $\mathcal{K}_1 \wedge \mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ by definition, the claim follows. The latter case is analogous. For the converse, we assume that there exists some $\mathcal{K}' \in \overline{\mathbf{TR}}_\alpha(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'$. By definition, this $\mathcal{K}'$ must be of the form $\mathcal{K}_1 \wedge \mathcal{K}''$ with $\mathcal{K}'' \in \overline{\mathbf{TR}}_{\alpha_1 \cdot \alpha}(\mathcal{K})$, or of the form $\dot{\neg}\mathcal{K}_1 \wedge \mathcal{K}''$ with $\mathcal{K}'' \in \overline{\mathbf{TR}}_{\alpha_2 \cdot \alpha}(\mathcal{K})$. In the former case, it follows from the induction hypothesis that $\mathcal{I} \models \mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1 \cdot \alpha}(\mathcal{K})$, and hence $\mathcal{I} \models \big(\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_1 \cdot \alpha}(\mathcal{K})\big) \vee \big(\dot{\neg}\mathcal{K}_1 \wedge \overline{\mathsf{TR}}_{\alpha_2 \cdot \alpha}(\mathcal{K})\big)$ and the claim follows. The second case, where $\mathcal{K}'$ is of the form $\dot{\neg}\mathcal{K}_1 \wedge \mathcal{K}''$, is analogous to the first one. □

Proof of Theorem 5. We employ the 3-Coloring problem for graphs. Assume a graph $G = (V, E)$ with $V = \{1, \dots, n\}$. We construct in polynomial time a KB $\mathcal{K}$ and an action α such that G is 3-colorable iff α 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:

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

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

(i) $\alpha_1 = (D \oplus \{o\}) \cdot (B_1 \oplus \{o\}) \cdots (B_n \oplus \{o\})$,

(ii) $\alpha_2^i = (B_i \ominus A_i^0) \cdot (B_i \ominus A_i^1) \cdot (B_i \ominus A_i^3)$, for all $i \in \{1, \dots, n\}$, and

(iii) $\alpha_3 = (D \ominus B_1) \cdots (D \ominus B_n)$.

Assume $\mathcal{I}$ is a model of $\mathcal{K}$ such that $S_\alpha(\mathcal{I}) \not\models \mathcal{K}$. It is possible to show that then G is 3-colorable.

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, $S_\alpha(\mathcal{I}) \not\models \mathcal{K}$. □

Proof of Theorem 6. 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. 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. As satisfiability testing in *DL-Lite*$_\mathcal{R}$ is feasible in polynomial time, we obtain 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.

Let P be the set of inclusions and assertions of $\mathcal{K}$. 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*

- $\mathcal{A} \subseteq \mathcal{A}^+$;
- for every assertion α, $\alpha \notin \mathcal{A}^+$ 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 a 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 $o_2 = o'_2$;

Figure 1: Completion for *DL-Lite*$^+_\mathcal{R}$ ABoxes

of $\mathcal{A}$ is a $\subseteq$-minimal set $\mathcal{A}^+$ of assertions that is closed under the conditions in Figure 1. 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. ☐

Proof of Theorem 7. The lower bound follows from Theorem 5, or alternatively, it can be proved by a reduction from finite unsatisfiability in *DL-Lite*$^+_\mathcal{R}$, employing the same reduction as in the proof of Theorem 4.

For the upper bound, assume a *DL-Lite*$^+_\mathcal{R}$ KB $\mathcal{K}$ and a simple action α. We proceed analogously to the Proof of 4. From Theorem 3 we know that α is not $\mathcal{K}$-preserving iff $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable. Moreover, we have shown that $\mathcal{K} \wedge \dot\neg\mathsf{TR}_{\alpha^*}(\mathcal{K})$ is finitely satisfiable iff there exists a $\mathcal{K}' \in \overline{\mathbf{TR}}_{\alpha^*}(\mathcal{K})$ such that $\mathcal{K} \wedge \mathcal{K}'$ is not finitely satisfiable, and $\mathcal{K}'$ can be obtained non-deterministically in polynomial time and is of size polynomial in α and $\mathcal{K}$. 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 α is simple, 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. ☐

Proof of Theorem 8. The proof is by reduction from the Halting problem. We reduce to (P1) and to (P2) deciding whether a deterministic Turing machine M accepts a word $w \in \{0, 1\}^*$.

For (P1), assume M is given by a tuple $M = (Q, \delta, q_0, q_a, q_r)$, where Q is a set of states, $\delta : \{0, 1, b\} \times Q \to \{0, 1, b\} \times Q \times \{+1, -1\}$ is the transition function, b is the blank symbol, $q_0 \in Q$ is the initial state, $q_a \in Q$ is the accepting state, and $q_r \in Q$ is the rejecting state. We can assume w.l.o.g. that after accepting or rejecting the input the machine returns the read/write head to the initial position.

Intuitively, we define an action that implements the effects of each possible transition from δ. We also have a pair of actions that "extend" the tape with blank symbols as needed. For the reduction we use the role $next$, concept names Sym_0, Sym_1, Sym_b, and St_q for each $q \in Q$.

The set Act of actions is defined as follows. For every $(\sigma, q) \in \{0, 1, b\} \times Q$ with $\delta(\sigma, q) = (\sigma', q', D)$ we have the action $\alpha_{\sigma,q} = (x_1, x_2) : next \wedge x_2 : Sym_\sigma \wedge x_2 : St_q \wedge (x_2, x_3) : next\,?\,(Sym_\sigma \ominus \{x_2\})(Sym_{\sigma'} \oplus \{x_2\})(St_q \ominus \{x_2\})(St_{q'} \oplus \{x_{2+D}\})$. To extend the tape with blank symbols, we have the actions α_r and α_l. In particular, $\alpha_r = x : (Sym_0 \sqcup Sym_1 \sqcup Sym_b) \wedge y : \neg(Sym_0 \sqcup Sym_1 \sqcup Sym_b)?(next \oplus \{(x, y)\})(Sym_b \oplus \{y\})\}$. The action α_l is obtained from α_r by replacing $(next \oplus \{(x, y)\})$ with $(next \oplus \{(y, x)\})$. We finally have an initialization action α_{init} which stores the initial configuration of M in the database. In particular, $\alpha_{init} = (a_1 : \neg(Sym_0 \sqcup Sym_1 \sqcup Sym_b))?(Sym_{\sigma_1} \oplus \{a_1\}) \cdots (Sym_{\sigma_m} \oplus \{a_m\})(St_{q_0} \oplus \{a_1\})$, where $\sigma_1 \cdots \sigma_m = w$. We let $\mathcal{K} = a_1 : St_{q_a} \sqcup St_{q_r}$ and the initial database $\mathcal{I}$ is empty, i.e. no domain element participates in a concept or a role.

It can be easily seen that the reduction is correct. If $\mathcal{K}$ has a plan, then M halts on w. Conversely, if M halts on w, then it halts within some number of steps s. One can verify that expanding the domain of $\mathcal{I}$ with s fresh elements is sufficient to find a plan for $\mathcal{K}$ using the actions in Act.

The above reduction also applies to (P2). It suffices to define a precondition KB $\mathcal{K}_{pre}$ that describes the above $\mathcal{I}$. Simply let $\mathcal{K}_{pre}$ be the conjunction of $(Sym_0 \sqcup Sym_1 \sqcup Sym_b \sqcup \exists next \sqcup \exists next^- \sqsubseteq \bot)$ and $\bigsqcup_{q \in Q} St_q \sqsubseteq \bot$. ☐

Proof of Theorem 9. The lower bound can be proven by an easy adaption of the reduction in Theorem 8.

For the upper bound we employ a non-deterministic polynomial space procedure that stores in memory a finite interpretation and non-deterministically applies actions until the goal is satisfied. Since the domain of each candidate interpretation is fixed and of size linear in the input, each of them can be represented in polynomial space. The number of possible interpretations is bounded by $c = 2^{r \cdot d^2 + c \cdot d}$, where r and c are respectively the number of concepts and roles appearing in the input set of actions, and d is the cardinality of the domain of the initial interpretation. Thus the procedure can be terminated after c many steps, without loss of completeness. We note that a counter that counts up to c can be implemented in polynomial space, and that model checking $\mathcal{ALCHOIQ}br$-formulae is feasible in polynomial space. ☐

Proof of Theorem 10. The lower bounds can be immediately inferred from the complexity of static verification with KBs in $\mathcal{ALCHOIQ}br$ (Theorem 4) and *DL-Lite*$^+_\mathcal{R}$ (Theorem 5).

For the upper bounds, we first guess a variable substitution σ and a sequence $P = \langle \alpha_1, \ldots, \alpha_m \rangle$ of at most k actions. By Theorem 2, it follows that P is a plan as desired iff $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1 \cdots \alpha_m}(\sigma(\mathcal{K}))$ is finitely satisfiable. To be able to check the finite satisfiability of $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1 \cdots \alpha_m}(\sigma(\mathcal{K}))$ within the desired bounds, we proceed similarly as above, and consider a procedure that non-deterministically builds a polynomial $\mathcal{K}'$ such that $\sigma(\mathcal{K}_{pre}) \wedge \mathcal{K}'$ is finitely satisfiable iff $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1 \cdots \alpha_m}(\sigma(\mathcal{K}))$ is finitely satisfiable. Note that the core difference between this proof and the ones of Theorems 4 and 7 is that now the formula $\mathsf{TR}_{\alpha_1 \cdots \alpha_m}(\sigma(\mathcal{K}))$ is not negated and hence, intuitively, we need to decide the existence of an interpretation that satisfies the negation of all formulas in $\overline{\mathbf{TR}}_\alpha(\mathcal{K})$, rather than satisfying just one of them.

We define a set of KBs $\mathbf{TR}^\wedge_\alpha(\mathcal{K})$ that is similar to $\overline{\mathbf{TR}}_\alpha(\mathcal{K})$, but contains the negation of the formulas in the latter, and uses conjunction rather than implications for the conditional axioms.

$$
\begin{aligned}
\mathbf{TR}^\wedge_\varepsilon(\mathcal{K}) =& \{\mathcal{K}\} \\
\mathbf{TR}^\wedge_{(A \oplus C) \cdot \alpha}(\mathcal{K}) =& \{\mathcal{K}'_{A \leftarrow A \sqcup C} \mid \mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})\} \\
\mathbf{TR}^\wedge_{(A \ominus C) \cdot \alpha}(\mathcal{K}) =& \{\mathcal{K}'_{A \leftarrow A \sqcap \neg C} \mid \mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})\} \\
\mathbf{TR}^\wedge_{(p \oplus r) \cdot \alpha}(\mathcal{K}) =& \{\mathcal{K}'_{p \leftarrow p \cup r} \mid \mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})\} \\
\mathbf{TR}^\wedge_{(p \ominus r) \cdot \alpha}(\mathcal{K}) =& \{\mathcal{K}'_{p \leftarrow p \setminus r} \mid \mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})\} \\
\mathbf{TR}^\wedge_{(\mathcal{K}_1\,?\,\alpha_1 [\![\alpha_2]\!]) \cdot \alpha}(\mathcal{K}) =& \{\mathcal{K}_1 \wedge \mathcal{K}' \mid \mathcal{K}' \in \mathbf{TR}^\wedge_{\alpha_1 \cdot \alpha}(\mathcal{K})\} \cup \{\dot{\neg}\mathcal{K}_1 \wedge \mathcal{K}' \mid \mathcal{K}' \in \mathbf{TR}^\wedge_{\alpha_2 \cdot \alpha}(\mathcal{K})\}
\end{aligned}
$$

Similarly as above, $|\mathbf{TR}^\wedge_\alpha(\mathcal{K})|$ may be exponential but each $\mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})$ is polynomial and can be built non-deterministically in polynomial time. We show below the following claim:

(‡) For every $\mathcal{I}$ and every $\mathcal{K}$, there exists some $\mathcal{K}' \in \mathbf{TR}^\wedge_\alpha(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_\alpha(\mathcal{K})$.

With (‡) we can easily show that $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ is finitely satisfiable iff there exists some $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ such that $\sigma(\mathcal{K}_{pre}) \wedge \mathcal{K}'$ is finitely satisfiable. For the 'only if' direction, assume $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ is finitely satisfiable. Then there exists some finite $\mathcal{I}$ such that $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$ and $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$. By (‡), for this $\mathcal{I}$ there is some $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ such that $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$. We choose this $\mathcal{K}'$. It follows that $\mathcal{I} \models \mathcal{K}'$ and, since $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$, we can conclude that $\sigma(\mathcal{K}_{pre}) \wedge \mathcal{K}'$ is finitely satisfiable. For the other direction, assume that there is no $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ such that $\sigma(\mathcal{K}_{pre}) \wedge \mathcal{K}'$ is finitely satisfiable. Then it follows that: $(*)$ $\mathcal{I} \not\models \mathcal{K}'$ for every $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ and every $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$. Assume towards a contradiction that $\sigma(\mathcal{K}_{pre}) \wedge \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ is satisfiable. Then there is some $\mathcal{I}$ with $\mathcal{I} \models \sigma(\mathcal{K}_{pre})$ and $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$, and by (‡), for this $\mathcal{I}$ there is some $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$ such that $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1\cdots\alpha_m}(\sigma(\mathcal{K}))$. This would imply that $\mathcal{I} \models \mathcal{K}'$, contradicting $(*)$. Having shown this, the upper bound follows directly from the complexity of deciding finite satisfiability of $\sigma(\mathcal{K}_{pre}) \wedge \mathcal{K}'$, and the fact that $\mathcal{K}'$ is of polynomial size and can be obtained non-deterministically in polynomial time.

It is only left to show (‡), what we do by induction on $s(\alpha)$. The base case is trivial, since for $\alpha = \varepsilon$ we have $\mathbf{TR}^{\wedge}_{\alpha}(\mathcal{K}) = \{\mathcal{K}\}$ and $\mathsf{TR}_{\alpha}(\mathcal{K}) = \mathcal{K}$, so we can set $\mathcal{K}' = \mathcal{K}$ and the claim follows.

For the case of $\alpha = A \oplus C \cdot \alpha'$, we have $\mathsf{TR}_{\alpha'}(\mathcal{K}) = \mathsf{TR}_{\alpha'}(\mathcal{K})$. By induction hypothesis there is some $\mathcal{K}'' \in \mathbf{TR}^{\wedge}_{\alpha'}(\mathcal{K})$ such that $\mathcal{I} \models \mathcal{K}''$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha'}(\mathcal{K})$. We let $\mathcal{K}' = \mathcal{K}''_{A \leftarrow A \sqcup C}$. Then $\mathcal{K}' \in \mathbf{TR}^{\wedge}_{\alpha}(\mathcal{K})$, and $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha'}(\mathcal{K})_{A \leftarrow A \sqcup C}$ as desired. The cases of $\alpha = (A \ominus C) \cdot \alpha'$, $\alpha = (p \oplus r) \cdot \alpha'$, and $\alpha = (p \ominus r) \cdot \alpha'$ are analogous.

Finally, if $\alpha = (\mathcal{K}_1 \mathbin{?} \alpha_1 [\![\alpha_2]\!]) \cdot \alpha'$, the choice of $\mathcal{K}'$ depends on $\mathcal{I}$. We distinguish two cases:

- If $\mathcal{I} \models \mathcal{K}_1$, let $\mathcal{K}'' \in \mathbf{TR}^{\wedge}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$ be such that $\mathcal{I} \models \mathcal{K}''$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$, which exists the induction hypothesis. Then we set $\mathcal{K}' = \mathcal{K}_1 \wedge \mathcal{K}''$. We have $\mathcal{K}'' \in \mathbf{TR}^{\wedge}_{\alpha}(\mathcal{K})$ by definition. Now we show that $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha}(\mathcal{K})$.
  Assume $\mathcal{I} \models \mathcal{K}'$. Then $\mathcal{I} \models \mathcal{K}''$, and $\mathcal{I} \models \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$. This ensures that $\mathcal{I} \models \dot{\neg}\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$. Since $\mathcal{I} \models \mathcal{K}_1$, we also have $\mathcal{I} \models \mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2 \cdot \alpha}(\mathcal{K})$. Since $\mathsf{TR}_{\alpha}(\mathcal{K}) = (\dot{\neg}\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})) \wedge (\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2 \cdot \alpha'}(\mathcal{K}))$, we obtain $\mathcal{I} \models \mathsf{TR}_{(\mathcal{K}_1 \mathbin{?} \alpha_1 [\![\alpha_2]\!]) \cdot \alpha'}(\mathcal{K})$ as desired.
  For the converse, assume $\mathcal{I} \models \mathsf{TR}_{\alpha}(\mathcal{K})$, that is, $\mathcal{I} \models \dot{\neg}\mathcal{K}_1 \vee \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$ and $\mathcal{I} \models \mathcal{K}_1 \vee \mathsf{TR}_{\alpha_2 \cdot \alpha'}(\mathcal{K})$. From the former and $\mathcal{I} \models \mathcal{K}_1$, it follows that $\mathcal{I} \models \mathsf{TR}_{\alpha_1 \cdot \alpha'}(\mathcal{K})$. By our selection of $\mathcal{K}''$, this implies $\mathcal{I} \models \mathcal{K}''$, and we also have that $\mathcal{I} \models \mathcal{K}_1$, we can conclude $\mathcal{I} \models \mathcal{K}'$ as desired.
- Otherwise, if $\mathcal{I} \models \dot{\neg}\mathcal{K}_1$, let $\mathcal{K}''$ be such that $\mathcal{K}'' \in \mathbf{TR}^{\wedge}_{\alpha_2 \cdot \alpha'}(\mathcal{K})$ and $\mathcal{I} \models \mathcal{K}''$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha_2 \cdot \alpha'}(\mathcal{K})$ (such a $\mathcal{K}''$ exists by the induction hypothesis), and let $\mathcal{K}' = \dot{\neg}\mathcal{K}_1 \wedge \mathcal{K}''$. Then $\mathcal{K}'' \in \mathbf{TR}^{\wedge}_{\alpha}(\mathcal{K})$, and the proof of $\mathcal{I} \models \mathcal{K}'$ iff $\mathcal{I} \models \mathsf{TR}_{\alpha}(\mathcal{K})$ is analogous to the first case.

□

Proof of Theorem 11. Problem (S) can be shown to be undecidable by employing the same reduction as for (P2) in Theorem 8. The coNExpTime lower bounds for (C) and ($\mathrm{S_b}$) trivially follow from finite satisfiability in $\mathcal{ALCHOIQ}br$.

For the upper bounds, we first observe that (C) reduces to validity testing in $\mathcal{ALCHOIQ}br$: an instance of (C) (as described above) is positive iff the formula $\sigma(\mathcal{K}'_{pre}) \rightarrow \mathsf{TR}_{\alpha_1\cdots\alpha_n}(\sigma(\mathcal{K}'))$ is valid, where $\mathcal{K}'_{pre}, \mathcal{K}'$ are obtained from $\mathcal{K}_{pre}, \mathcal{K}$ by replacing every variable by a fresh individual. Deciding validity of $\sigma(\mathcal{K}'_{pre}) \rightarrow \mathsf{TR}_{\alpha_1\cdots\alpha_n}(\sigma(\mathcal{K}'))$ in turn reduces to deciding whether $\sigma(\mathcal{K}'_{pre}) \wedge \dot{\neg}\mathsf{TR}_{\alpha_1\cdots\alpha_n}(\sigma(\mathcal{K}'))$ is finitely unsatisfiable. The upper bounds for (C) then follow from the NP and NExpTime upperbounds for the satisfiability of KBs of the form $\mathcal{K}' \wedge \dot{\neg}\mathsf{TR}_{\alpha}(\mathcal{K})$ shown in the proofs of Theorems 4 and 7.

Negative instances of ($\mathrm{S_b}$), where $\mathcal{K}_{pre}$ is the precondition and $\mathcal{K}$ is the goal, can be recognized in NExpTime. Such a test comprises building an exponentially large set of all candidate action sequences of length at most k, and then making sure that that each candidate is invalidated. That is, each candidate action sequence P induces an instance of (C), which can be shown negative in NExpTime. In the case of *DL-Lite*$^{+}_{\mathcal{R}}$ and simple actions, we can guess non-deterministically a sequence of actions of length at most k and then check that the induced instance of (C) is positive, which is a test in coNP. It is not difficult to see that the $\mathrm{NP}^{\mathrm{NP}}$ upper bound is tight. This can be shown by a polynomial time reduction from evaluating QBFs of the form $\gamma = \exists p_1 \ldots \exists p_n \forall q_1 \ldots \forall q_m.\psi$, where ψ is a Boolean combination over propositional variables

$V = \{p_1, \ldots, p_n, q_1, \ldots, q_m\}$. We can assume that negation in ψ occurs in front of propositional variables only. For the reduction to ($\mathsf{S_b}$), we employ concept names T and F, and individual names o_v for each propositional variable $v \in V$. We let $\mathcal{K}_{pre} = \big(\bigwedge_{1 \le i \le n} o_{p_i} : \neg(T \sqcup F)\big) \wedge \big(\bigwedge_{1 \le i \le m} o_{q_i} : (T \sqcup F) \sqcap (\neg T \sqcup \neg F)\big)$. Intuitively, each initial interpretation encodes an assignment for the variables $q_1, \ldots, q_m$, but does not say anything about $p_1, \ldots, p_n$. The latter is determined by choosing a candidate plan. To this end, for each $1 \le i \le n$, we construct the following actions:

$$\alpha_i = o_{p_i} : \neg F \,?\, T \oplus \{o_{p_i}\}, \qquad \alpha_i' = o_{p_i} : \neg T \,?\, F \oplus \{o_{p_i}\}.$$

We finally let $k = n$ and let $\mathcal{K}$ be the KB obtained from ψ by replacing each negative literal $\neg v$ by $o_v : F$ and each positive literal v by $o_v : T$. It is not difficult to see that γ evaluates to *true* iff the constructed instance of ($\mathsf{S_b}$) is positive. □