Verification of Evolving Graph-structured Data under Expressive Path Constraints

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

1 Free University of Bozen-Bolzano, Bozen, Italy
2 TU Wien, Viena, Austria

Abstract

Integrity constraints play a central role in databases and, among other applications, are fundamental for preserving data integrity when databases evolve as a result of operations manipulating the data. In this context, an important task is that of static verification, which consists in deciding whether a given set of constraints is preserved after the execution of a given sequence of operations, for every possible database satisfying the initial constraints. In this paper, we consider constraints over graph-structured data formulated in an expressive Description Logic (DL) that allows for regular expressions over binary relations and their inverses, generalizing many of the well-known path constraint languages proposed for semi-structured data in the last two decades. In this setting, we study the problem of static verification, for operations expressed in a simple yet flexible language built from additions and deletions of complex DL expressions. We establish undecidability of the general setting, and identify suitable restricted fragments for which we obtain tight complexity results, building on techniques developed in our previous work for simpler DLs. As a by-product, we obtain new (un)decidability results for the implication problem of path constraints, and improve previous upper bounds on the complexity of the problem.

1998 ACM Subject Classification H.2 [Database Management] General

Keywords and phrases Path constraints, Description Logics, Graph databases, Static verification

Digital Object Identifier 10.4230/LIPIcs.ICDT.2016.15

1 Introduction

Integrity constraints play a central role in databases and, among many other applications, are fundamental for preserving data integrity when databases evolve as a result of operations manipulating the data [1, 23, 6]. A fundamental problem in this context is *static verification*: given a set of integrity constraints, and a sequence of operations that describe changes on databases (over the same schema), the goal is to verify whether the constraints are *preserved* by the operations, that is, they are satisfied after their application, for *every* database that initially satisfies the constraints. This allows one to establish the acceptability of sequences of operations, which guarantees that applications maintain data integrity at runtime, independently of the specific database states that may be reached. However, static verification is very hard, and identifying sufficiently expressive languages for integrity constraints and data operations that allow for decidable verification is challenging.

In this paper, we consider *graph-structured data* (GSD), that is, relational data that contains unary and binary relations only, and thus admits a natural representation as a labeled graph. This data model is well suited for those settings where the data does not comply to a fixed schema, and the *topology* of the data relations is central. The GSD model became important already two decades ago due to the close relationship with semi-structured data [2, 13]. In the last decade it has gained renewed interest due to its relevance in the

Semantic Web and a diverse range of areas, including social networks, life-sciences, and program analysis, see e.g., [5, 36] and references therein. The study of query languages for GSD has been the focus of extensive research efforts in the database community over the last decade, based on the common consensus that GSD requires *navigational* query languages that allow, as minimal required functionality, to extract nodes that are connected by paths complying with a given regular language [36, 30]. The exploration of constraints for this data model is somehow more limited, partly due to the fact that even simple formalisms for constraining relations between regular paths result in undecidability of basic inference problems. Indeed, early proposals for path constraint languages are still viewed as adequate [3, 14, 16, 26, 17], but their wider adoption is hindered by the fact their implication problem is only known to be decidable under very strong restrictions. Recently there has been much interest in the study of containment for expressive query languages for GSD [27, 20]. In this basic form of static analysis, queries can be seen as expressing constraints over GSD, but also here inference turns out to be undecidable unless severe restrictions are imposed. For example, containment of *Graph-XPath* queries, a variation of XPath advocated for querying GSD, is undecidable in general, and has been shown decidable only when restricted to the so-called *path-positive* fragment [27]. In this paper we show undecidability for a path constraint language that is significantly more restricted than Graph-XPath, and even than the path constraints in [14], and improve the previous undecidability results that required to express paths that return to the initial point [14, 27].

We advocate an expressive Description Logic (DL) as constraint language for GSD. DLs are a family of languages tailored for representing structured knowledge, and for supporting inference over it [8]. They formalize domain knowledge by describing complex classes of objects, called *concepts*, and binary relations between them, called *roles*. Most DLs can be seen as (syntactic variants of) decidable fragments of classical first-order (FO) logic, or its extension with transitive closure. Different DLs provide different expressive means to describe knowledge, with the computational complexity of inference varying accordingly. DLs are the basis of state-of-the-art *ontology languages* for sharing domain conceptualizations [9], and they are naturally suited for describing data sources. They have been applied for the static analysis of traditional data models, such as UML class diagrams [10] and Entity Relationship schemata [7]. In the paradigm of *ontology based data access* [34, 28], which has gained great importance in the last decade, DL ontologies are used to describe possibly heterogeneous data sources, facilitating their management, and leveraging domain knowledge to improve access to them. Query answering and containment in this setting has been extensively studied, for a range of DLs and query languages [11, 15, 33].

In this paper we show that DLs are adequate also as constraint languages for GSD. We focus on the expressive DL $\mathcal{ZOI}$, also known as $\mathcal{ALCOIb}^{\mathsf{Self}}_{reg}$ [18], which features regular expressions over binary relations and their inverses, and allows for using them to impose complex relations between concepts and roles. $\mathcal{ZOI}$ can express the full path-positive fragment of Graph-XPath, and supports additional features that allow it to express even richer constraints on GSD. We also show that well-known path constraint languages proposed for GSD in the past [3, 14] can be naturally expressed in (variations of) $\mathcal{ZOI}$. Moreover, we can leverage results from the DL community to generalize and improve previous upper bounds on the complexity of the implication problem for decidable fragments of path constraints. This requires, however, to lift existing algorithms and complexity results for reasoning in $\mathcal{ZOI}$ to *finite* structures, since in the setting of verification of evolving GSD we are interested in *finite data instances*, and so far this DL had been studied over unrestricted, possibly infinite, models only [18]. This transfer of results to the finite setting is fortunately possible since $\mathcal{ZOI}$ enjoys *finite model property* (FMP), as we are able to prove. This is a crucial stepping stone for our results, and an interesting contribution on its own right. Indeed, while the FMP has been long known for the closely related *converse PDL* [24, 21], $\mathcal{ZOI}$ has several further features. In particular, it allows for Boolean combinations of roles that make standard filtration techniques not directly applicable, and call for more subtle arguments that borrow ideas from FMP proofs for the two-variable fragment of first-order logic [32, 25].

For expressing operations on GSD we use the action language proposed in [4], that allows for the (possibly conditional) composition of basic operations that add or delete from a predicate the objects selected by a complex DL concept or role.[1] The undecidability of general path constraint implication implies that static verification becomes undecidable when if complex roles are allowed in actions. However, we regain decidability by restricting the roles in the action to be *simple roles* that allow for union, intersection, and difference of possibly inverse roles, but disallow composition and the Kleene star. Under these restrictions, we can rely on the techniques of [4] to reduce static verification in the presence of $\mathcal{ZOI}$ constraints to satisfiability of $\mathcal{ZOI}$ KBs, obtaining a tight ExpTime upper bound for the former.

The paper is organized as follows. In Section 2 we introduce the DL $\mathcal{ZOI}$, and establish the finite model property for it. Section 3 is devoted to path constraint languages. We tighten the undecidability of path constraint implication shown in [14]. We also generalize the decidability in [3] to a richer class, and improve the upper bound from 2ExpSpace to ExpTime by reducing the problem to reasoning in $\mathcal{ZOI}$. Section 4 studies the static verification of $\mathcal{ZOI}$ constraints over GSD, for actions expressed in the language proposed in [4]. We show that the problem is undecidable if arbitrary $\mathcal{ZOI}$ roles occur in actions, and impose suitable restrictions to obtain ExpTime decidability using the techniques of [4].

2 Expressive DLs for Expressing Constraints over GSD

In this paper, we formalize GSD as relational structures, which we call *instances*, over a unary and binary relational signature. We propose to use the rich DL $\mathcal{ZOI}$, also known as $\mathcal{ALCOIb}^{\mathsf{Self}}_{reg}$ [18], to express constraints over graph structured data. This is natural as, like other DLs, $\mathcal{ZOI}$ is defined over a relational vocabulary that contains unary and binary predicates only (respectively called *concept names* and *role names* in DL jargon) and the structures over which is interpreted are precisely GSD instances. A distinguishing feature of $\mathcal{ZOI}$, which makes it especially adequate for describing GSD, is that it can express relations between objects by allowing for *complex roles* defined using regular expressions.

▶ **Definition 1** ($\mathcal{ZOI}$ syntax)**.** We consider fixed, countably infinite sets $\mathsf{N_C}$ of *concept names*, $\mathsf{N_R}$ of *role names*, and $\mathsf{N_I}$ of individual names. We assume that the set $\mathsf{N_C}$ contains the special concepts $\top$ (top) and $\bot$ (bottom), while $\mathsf{N_R}$ contains the top (universal) role T and the bottom (empty) role B. We define ($\mathcal{ZOI}$) *atomic concepts* B, *concepts* C, C', *atomic roles* P, *simple roles* S, S', and *roles* R, R', where $a, b \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $r \neq \mathsf{T}$, according to the following syntax:

$$\begin{array}{rcl} B & \longrightarrow & A \mid \{a\} \\ C, C' & \longrightarrow & B \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \\ & & \forall R.C \mid \exists R.C \mid \exists S.\mathsf{Self} \end{array} \qquad \begin{array}{rcl} P & \longrightarrow & r \mid r^{-} \mid \{(a,b)\} \\ S, S' & \longrightarrow & P \mid S \cap S' \mid S \cup S' \mid S \setminus S' \\ R, R' & \longrightarrow & \mathsf{T} \mid \varepsilon \mid id(C) \mid S \mid R \cup R' \mid \\ & & R \circ R' \mid R^{*} \end{array}$$

We call $\mathcal{ZOI}$ expressions $\{a\}$ *nominal concepts*, and $\{(a,b)\}$ *nominal roles*. ◀

[1] In our setting, updates are performed data that is viewed as complete, and DL constraints are not used to infer new knowledge. Thus we do not run in the expressiveness issues considered e.g., in [31, 19].

We use $\mathcal{ZOI}$ concepts and roles to define a general form of knowledge bases, in which we allow for Boolean combinations of intensional and extensional level statements.

▶ **Definition 2** ($\mathcal{ZOI}$ knowledge bases)**.** A *concept inclusion* is an expression of the form $C \sqsubseteq C'$, where C, C' are arbitrary concepts, and a *role inclusion* is an expression of the form $S \sqsubseteq S'$, where S, S' are simple roles. An *assertion* is an expression of the form $C(a)$, $S(a, a')$, or $a \neq a'$, where C is a concept, S a simple role, and $\{a, a'\} \subseteq \mathsf{N_I}$. Then, *($\mathcal{ZOI}$) knowledge bases* (KBs) are defined inductively as follows:

(i) every inclusion and every assertion is a KB;
(ii) if $\mathcal{K}$, $\mathcal{K}'$ are KBs, so are $\mathcal{K} \wedge \mathcal{K}'$, $\mathcal{K} \vee \mathcal{K}'$, and $\neg\mathcal{K}$. ◀

The semantics of $\mathcal{ZOI}$ is based on standard relational structures. An *instance* (or *interpretation*) $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps each individual $a \in \mathsf{N_I}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each concept name $A \in \mathsf{N_C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each role name $r \in \mathsf{N_R}$ to a set $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, in such a way that $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$, $\mathsf{T}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and $\mathsf{B}^{\mathcal{I}} = \emptyset$. The function $\cdot^{\mathcal{I}}$ is inductively extended to all $\mathcal{ZOI}$ concepts and roles as follows:

$$\begin{aligned}
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} & \{(a,b)\}^{\mathcal{I}} &= \{(a^{\mathcal{I}}, b^{\mathcal{I}})\}\\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (r^-)^{\mathcal{I}} &= \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}\\
(C \sqcap C')^{\mathcal{I}} &= C^{\mathcal{I}} \cap C'^{\mathcal{I}} & (S \setminus S')^{\mathcal{I}} &= S^{\mathcal{I}} \setminus S'^{\mathcal{I}}\\
(C \sqcup C')^{\mathcal{I}} &= C^{\mathcal{I}} \cup C'^{\mathcal{I}} & (S \cap S')^{\mathcal{I}} &= S^{\mathcal{I}} \cap S'^{\mathcal{I}}\\
(\forall R.C)^{\mathcal{I}} &= \{x \mid \forall y.(x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\} & (R \cup R')^{\mathcal{I}} &= R^{\mathcal{I}} \cup R'^{\mathcal{I}}\\
(\exists R.C)^{\mathcal{I}} &= \{x \mid \exists y.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} & (R \circ R')^{\mathcal{I}} &= R^{\mathcal{I}} \circ R'^{\mathcal{I}}\\
(\exists S.\mathrm{Self})^{\mathcal{I}} &= \{x \mid (x,x) \in S^{\mathcal{I}}\} & (R^*)^{\mathcal{I}} &= (R^{\mathcal{I}})^*\\
& & (id(C))^{\mathcal{I}} &= \{(x,x) \mid x \in C^{\mathcal{I}}\}\\
& & (\varepsilon)^{\mathcal{I}} &= \{(x,x) \mid x \in \Delta^{\mathcal{I}}\}
\end{aligned}$$

where $\cap$, $\cup$, and $\setminus$ are overloaded to denote also the standard set-theoretic operations, $\circ$ to denote composition, and $\cdot^*$ to denote the reflexive transitive closure of a binary relation.

$\mathcal{I}$ *satisfies* the inclusion $E \sqsubseteq E'$ if $E^{\mathcal{I}} \subseteq E'^{\mathcal{I}}$, the assertions $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, $S(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in S^{\mathcal{I}}$, and $a \neq b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. Satisfaction is extended in the usual way to KBs, which are Boolean combinations of inclusions and assertions. When $\mathcal{I}$ satisfies $\mathcal{K}$, we also say that $\mathcal{I}$ is a *model* of $\mathcal{K}$, and denote it with $\mathcal{I} \models \mathcal{K}$.

As basic reasoning task we consider *KB satisfiability*, which consists in deciding, given a KB $\mathcal{K}$, whether $\mathcal{K}$ admits a model. Other standard reasoning tasks, like concept (resp., role) satisfiability, that is, deciding whether there exists an interpretation where the extension of a given concept (resp., role) is not empty, can be reduced to KB satisfiability.

▶ Remark. The roles ε and $\{(a,b)\}$, which are not usually included in $\mathcal{ZOI}$, are just syntactic sugar. Indeed, ε has the same meaning as $id(\top)$ and captures the identity relation. Nominal roles $\{(a,b)\}$ can be easily simulated in $\mathcal{K}$ by replacing each occurrence of $\{(a,b)\}$ by a fresh role name r_{ab}, and conjunctively adding to $\mathcal{K}$ the KB $r_{ab}(a,b) \wedge (\exists r_{ab}.\top \sqsubseteq \{a\}) \wedge (\top \sqsubseteq \forall r_{a,b}.\{b\})$. This ensures that $r_{ab}^{\mathcal{I}} = \{(a^{\mathcal{I}}, b^{\mathcal{I}})\}$ in every model of the modified KB.

▶ **Example 3.** As a running example, we consider the following self-explanatory instance $\mathcal{I}_{\mathsf{Uni}}$. For simplicity, in the examples we interpret individuals as themselves (i.e., we make the standard name assumption).

$$\begin{array}{rcl}
\mathsf{Dept}^{\mathcal{I}} & = & \{\text{CS\_Dept}\}\\
\mathsf{Program}^{\mathcal{I}} & = & \{\text{BSc\_CSci}, \text{MSc\_CompLogic}, \text{MSc\_Bioinformatics}\}\\
\mathsf{Course}^{\mathcal{I}} & = & \{\text{DataStruct:CS202}, \text{FoundDBs:CS327}, \text{DLs:CS451}\}\\
\mathsf{partOf}^{\mathcal{I}} & = & \{(\text{DLs:CS451}, \text{mod\_KR})\}\\
\mathsf{offers}^{\mathcal{I}} & = & \{(\text{CS\_Dept}, \text{BSc\_CompSci}), (\text{CS\_Dept}, \text{MSc\_CompLogic}),\\
& & (\text{CS\_Dept}, \text{MSc\_Bioinformatics})\{(\text{CS\_Dept}, \text{DataStruct:CS202}),\\
& & (\text{CS\_Dept}, \text{FoundDBs:CS327}), (\text{CS\_Dept}, \text{DLs:CS451})\}
\end{array}$$

$$\begin{array}{rcl}
\mathsf{requires}^{\mathcal{I}} & = & \{(\text{BSc\_CSci}, \text{DataStruct:CS202}), (\text{MSc\_CompLogic}, \text{FoundDBs:CS327}),\\
& & (\text{MSc\_CompLogic}, \text{mod\_KR}), (\text{MSc\_Bioinformatics}, \text{DLs:CS451})\}
\end{array}$$

Consider the KB $\mathcal{K}_{\mathsf{Uni}}$ defined as the conjunction of the following $\mathcal{ZOI}$ constraints: ϕ_1 says that the domain of 'offers' are the departments, and ϕ_2 says that its range is the union of programs and courses. Similarly, ϕ_3 and ϕ_4 restrict the domain of 'requires' to programs, and its range to courses other entities that comprise courses, like modules. Finally ϕ_5 says that every course that is required (directly, or because it is part of a required module) must be offered.

$$\begin{array}{llll}
\phi_1 = & \exists\mathsf{offers}.\top \sqsubseteq \mathsf{Dept} & \phi_2 = & \top \sqsubseteq \forall\mathsf{offers}.(\mathsf{Program} \sqcup \mathsf{Course})\\
\phi_3 = & \exists\mathsf{requires}.\top \sqsubseteq \mathsf{Program} & \phi_4 = & \top \sqsubseteq \forall\mathsf{requires}.(\exists\mathsf{partOf}^{-*}.\mathsf{Course})\\
\phi_5 = & \mathsf{Course} \sqcap \exists(\mathsf{partOf}^{*} \circ \mathsf{requires}^{-}).\top \sqsubseteq \exists\mathsf{offers}^{-}.\top & &
\end{array}$$

Note that all these constraints are satisfied by our instance, that is, $\mathcal{I}_{\mathsf{Uni}} \models \mathcal{K}_{\mathsf{Uni}}$.

We note that $\mathcal{ZOI}$ is closely related to *path-positive Graph-XPath* (abbreviated $\mathsf{GXPath}_{reg}^{path\text{-}pos}$) introduced in [30]. By viewing arc labels as role names, node formulas in $\mathsf{GXPath}_{reg}^{path\text{-}pos}$ can be written as $\mathcal{ZOI}$ concepts, and $\mathsf{GXPath}_{reg}^{path\text{-}pos}$ path formulas as $\mathcal{ZOI}$ roles. Additionally $\mathcal{ZOI}$ extends $\mathsf{GXPath}_{reg}^{path\text{-}pos}$ with other features, such as Boolean combinations of node and path labels (i.e., Boolean concepts and roles), nominals, and concepts of the form $\exists S.\mathrm{Self}$.

In [18], a tree-automata based algorithm for checking satisfiability of $\mathcal{ZOI}$ concepts is provided, and by using a variant of *internalization* [35], this is exploited to check satisfiability of $\mathcal{ZOI}$ KBs constituted by a conjunction of (positive) assertions and inclusions. It is easy to extend internalization also to $\mathcal{ZOI}$ KBs of the more general form considered here, and thus reduce satisfiability of a $\mathcal{ZOI}$ KB to satisfiability of a $\mathcal{ZOI}$ concept. The proof is given in the extended version of the paper.

▶ **Theorem 4.** *Given a $\mathcal{ZOI}$ KB $\mathcal{K}$, one can construct in linear time a $\mathcal{ZOI}$ concept $C_{\mathcal{K}}$ such that $\mathcal{K}$ is satisfiable if and only if $C_{\mathcal{K}}$ is so.*

From this result and the ExpTime upper bound for concept satisfiability given in [18], it follows immediately that satisfiability of $\mathcal{ZOI}$ KBs is decidable in single exponential time. This is worst-case optimal, since the problem is ExpTime-hard even for significantly simpler DLs like $\mathcal{ALC}$ [8].

▶ **Theorem 5** ([18]). *Checking satisfiability of $\mathcal{ZOI}$ KBs is an ExpTime-complete problem.*

Finite Model Reasoning in $\mathcal{ZOI}$. In the setting of GSD we are usually interested in finite instances. In the DL literature, however, finite model reasoning has received significantly less attention than reasoning with respect to unrestricted models. To our knowledge, finite model reasoning for $\mathcal{ZOI}$ has not been addressed so far. However, as we show in the following, $\mathcal{ZOI}$ enjoys the *finite model property*, which states that every satisfiable KB admits a model whose domain is finite. In line with what has been done for other logics that cannot express functionality, keys, or number restrictions, we can show this through a filtration argument [24, 21]. However, due to the presence of both transitive closure over roles and role intersection and difference, the proof is more involved than for logics that involve none or only one of the two kinds of constructs.

We say a KB $\mathcal{K}$ is *finitely satisfiable* if it admits a finite model, i.e., a model with a finite domain. The proof of the following result is given in the extended version of the paper.

▶ **Theorem 6.** *Let $\mathcal{K}$ be a $\mathcal{ZOI}$ KB. Then $\mathcal{K}$ is satisfiable if and only if $\mathcal{K}$ is finitely satisfiable.*

3 Path Constraints

We define a language for path constraints inspired by [3, 14, 26] and closely related to $\mathcal{ZOI}$.

▶ **Definition 7** (Path constraints). A *path constraint* φ has the form $[R_p](R_\ell \subseteq R_r)$, where R_p, R_ℓ, and R_r are arbitrary $\mathcal{ZOI}$ roles. The role R_p is called *prefix* of φ, while the roles R_ℓ and R_r are respectively called the *left tail* and the *right tail* of φ. If $R_p = \varepsilon$, we call φ a *prefix-empty* constraint[2], and write it simply as $R_\ell \subseteq R_r$. ◀

This definition generalizes the well-known path constraint languages from [3, 14]. A complex role R built from the symbols in $\mathsf{N_R} \cup \{\varepsilon\}$ using $\circ$, $\cup$, and * is called a *(one-way) regular path role*, and if additionally it does not contain $\cup$ or * then it is called a *(one-way) word role*. A *one-way regular path constraint* (called simply *path constraint* in [3]) is a prefix-empty constraint where R_ℓ and R_r are one-way regular path roles. If, in addition, R_ℓ and R_r are one-way word roles, the path constraint is called a *word constraint* in [3]. The language of path constraints in [14] allows for non-empty prefixes, but restricts the left tail to be a one-way word role, and the right tail to be either a one-way word role (in the so-called *forward constraints*), or an *inverted one-way word role* (in *backward constraints*), which is a sequence of concatenated inverses of role names (that is, $r_1^- \circ \cdots \circ r_n^-$ with $n \geq 0$).[3]

Now we define the semantics of path constraints and their fundamental reasoning problem, namely *implication* of path constraints, both in its finite and in its unrestricted variants.

In the semantics of early path constraint languages [3, 14], every instance has a distinguished root object at which the constraints are enforced. We introduce a minor variation of this semantics, which we call *pointed* semantics, where rather than a fixed name for the root node, we allow for any individual name to be used as its identifier. Later works advocated what we call the *global* semantics [22, 26], in which constraints are enforced at every point in the model, rather than at just one. We note that the global semantics is in general computationally more costly, and causes undecidability of the implication problem for some fragments that are decidable under the pointed semantics [3, 14]. We discuss below how both

[2] *Prefix-empty* constraints were called *simple* in [14]. We use a different name to avoid confusion with the *simple roles* of Definition 1.
[3] We note that [14] uses a different syntax with explicit variables.

semantics can be naturally captured in DLs, and provide decidability and undecidability results for both of them.

▶ **Definition 8** (Pointed and rooted semantics, implication problem)**.** Let $\varphi = [R_p](R_\ell \subseteq R_r)$ be a path constraint. For an interpretation $\mathcal{I}$, we let $\varphi^\mathcal{I}$ be the set of objects $d \in \Delta^\mathcal{I}$ such that for each $d', d'' \in \Delta^\mathcal{I}$, if $(d, d') \in R_p^\mathcal{I}$ and $(d', d'') \in R_\ell^\mathcal{I}$, then $(d', d'') \in R_r^\mathcal{I}$.

A *pointed instance* is a pair $\mathcal{I}, a$ of an instance $\mathcal{I}$ and an individual a. We call $\mathcal{I}, a$ a *pointed model* of φ, and write $\mathcal{I}, a \models \varphi$ if $a^\mathcal{I} \in \varphi^\mathcal{I}$. Similarly, we write $\mathcal{I}, a \models \Gamma$ for a set Γ of constraints, if $\mathcal{I}, a \models \varphi$ for each $\varphi \in \Gamma$. We write $\Gamma, a \models \varphi$ if $\mathcal{I}, a \models \varphi$ for every pointed model $\mathcal{I}, a$ of Γ, and write $\Gamma, a \models_{\mathit{fin}} \varphi$ if $\mathcal{I}, a \models \varphi$ for every *finite* pointed model $\mathcal{I}, a$ of Γ. The *(finite) pointed implication problem* consists in deciding, given an individual a, a set Γ of path constraints, and a path constraint φ, whether $\Gamma, a \models_{(\mathit{fin})} \varphi$.

Let $\varphi = [R_p](R_\ell \subseteq R_r)$ be a path constraint. We call $\mathcal{I}$ a *global model* of φ, and write $\mathcal{I} \models \varphi$ if $\varphi^\mathcal{I} = \Delta^\mathcal{I}$. We write $\mathcal{I} \models \Gamma$ for a set Γ of constraints, if $\mathcal{I} \models \varphi$ for each $\varphi \in \Gamma$. We write $\Gamma \models \varphi$ if $\mathcal{I} \models \varphi$ for every $\mathcal{I}$ with $\mathcal{I} \models \Gamma$, and write $\Gamma \models_{\mathit{fin}} \varphi$ if $\mathcal{I} \models \varphi$ for every *finite* $\mathcal{I}$ with $\mathcal{I} \models \Gamma$. The *(finite) global implication problem* consists in deciding, given a set Γ of path constraints and a path constraint φ, whether $\Gamma \models_{(\mathit{fin})} \varphi$. ◀

▶ **Example 9.** Consider the constraint $\varphi_1 = R_1 \subseteq R_2$, where
$R_1 = \mathit{id}(\mathsf{Dept}) \circ \mathsf{partOf}^{-*} \circ \mathsf{offers} \circ \mathsf{requires}^{-} \circ \mathsf{partOf}^{-*}$
$R_2 = \mathit{id}(\mathsf{Dept}) \circ \mathsf{partOf}^{-*} \circ \mathsf{offers}$
Intuitively, (a node interpreting) a department satisfies φ_1 if every course required by a program offered by (a suborganization of) the department is offered by (a suborganization of) the same department. With the rooted semantics, we can enforce the constraint for some specific departments. For example, we may require it for computer science, and our example instance satisfies it: $\mathcal{I}_{\mathsf{Uni}}, \mathrm{CS\_Dept} \models \varphi_1$. With the global semantics, the constraint would apply to all departments, but we can easily modify it so that it applies only to the desired departments, e.g., $(\mathit{id}(\mathrm{CS\_dept}) \cdot R_1) \subseteq R_2$.

To illustrate the use of prefixes, suppose that the policy expressed by φ_1 is enforced not at the department level, but at the higher faculty/school level. For example, suppose that the School of Science and Engineering requires that all departments offer within their department every mandatory course in their programs (while other schools may allow for mandatory courses that are offered by different departments). This is captured by the constraint with non-empty prefix $\varphi_1 = [\mathit{id}(\mathsf{School\_SciEng}) \circ \mathsf{hasDepartment}](R_1 \subseteq R_2)$.

Expressing Path Constraints in $\mathcal{ZOI}$ with Role Difference. To express path constraints, we extend $\mathcal{ZOI}$ with difference $R \setminus R'$ of arbitrary roles, resulting in the logic we call $\mathcal{ZOI}^\setminus$.

▶ **Definition 10.** $\mathcal{ZOI}^\setminus$ roles are defined analogously to $\mathcal{ZOI}$ roles, except that for complex roles we have $R, R' \longrightarrow \top \mid \mathit{id}(C) \mid S \mid R \cup R' \mid R \circ R' \mid R \setminus R' \mid R^*$.
The syntax and semantics of $\mathcal{ZOI}^\setminus$ concepts, assertions, axioms, and knowledge bases are defined as for $\mathcal{ZOI}$, but allowing for $\mathcal{ZOI}^\setminus$ roles in the place of $\mathcal{ZOI}$ roles. ◀

Entailment of path constraints defined above can be reduced to reasoning in $\mathcal{ZOI}^\setminus$:

▶ **Lemma 11.** *For a path constraint $\varphi = [R_p](R_\ell \subseteq R_r)$, let $C_\varphi = \forall R_p.(\forall(R_\ell \setminus R_r).\bot)$. Then, for every instance $\mathcal{I}$, we have that $\varphi^\mathcal{I} = C_\varphi^\mathcal{I}$. Consequently, for a set Γ of path constraints, a* *path constraint φ, and $a \in \mathsf{N_I}$, we have:*

$$\begin{array}{lll}
\Gamma, a \models \varphi & \textit{iff} & \big(\bigsqcap_{\gamma\in\Gamma} C_\gamma \sqcap \neg C_\varphi\big)(a) \quad \textit{is unsatisfiable,}\\
\Gamma, a \models_{\mathit{fin}} \varphi & \textit{iff} & \big(\bigsqcap_{\gamma\in\Gamma} C_\gamma \sqcap \neg C_\varphi\big)(a) \quad \textit{is finitely unsatisfiable,}\\
\Gamma \models \varphi & \textit{iff} & \bigwedge_{\gamma\in\Gamma}(\top \sqsubseteq C_\gamma) \wedge \dot\neg(\top \sqsubseteq C_\varphi) \quad \textit{is unsatisfiable,}\\
\Gamma \models_{\mathit{fin}} \varphi & \textit{iff} & \bigwedge_{\gamma\in\Gamma}(\top \sqsubseteq C_\gamma) \wedge \dot\neg(\top \sqsubseteq C_\varphi) \quad \textit{is finitely unsatisfiable.}
\end{array}$$

We will see that, unfortunately, both implication of path constraints and reasoning in $\mathcal{ZOI}^\setminus$ are undecidable. Before moving to these negative results, though, we point out that the lemma above implies that the upper bounds for reasoning in plain $\mathcal{ZOI}$ extend to the implication of path constraints γ where only simple $\mathcal{ZOI}$ roles occur. Since in this case the resulting C_γ is a $\mathcal{ZOI}$ concept, from Lemma 11 and theorem 5 we get:

▶ **Corollary 12.** *Let Γ be a set of path constraints and φ a path constraint such that, for each $\gamma = [R_p](R_\ell \subseteq R_r) \in \Gamma \cup \{\varphi\}$, R_p, R_ℓ and R_r are all simple $\mathcal{ZOI}$ roles. Then $\Gamma, a \models \varphi$, $\Gamma, a \models_{\mathit{fin}} \varphi$, $\Gamma \models \varphi$, and $\Gamma \models_{\mathit{fin}} \varphi$ are all decidable in* ExpTime.

Undecidability of Path Constraint Implication. Unfortunately, both the finite and the unrestricted implication problems are undecidable in rather restricted settings. The following result was established already several years ago:[4]

▶ **Theorem 13** ([14]). *Assume that every instance has a distinguished* root *element o, and that there is some $a_o \in \mathsf{N_I}$ such that $a_o^\mathcal{I} = o$ in every $\mathcal{I}$. The problem of checking whether $\Gamma, a_o \models \varphi$ and whether $\Gamma, a_o \models_{\mathit{fin}} \varphi$ are undecidable, even when φ and all constraints in Γ satisfy one of the following two restrictions:*

- *All prefixes, left tails, and right tails are one-way word roles (that is, only forward constraints according to [14] are allowed).*
- *All prefixes and left tails are one-way word roles different from ε, and each right tail is a one-way word role or an inverted one-way word role, and is different from ε.*

We strengthen this result, showing undecidability when both restrictions apply: only one-way word roles of length one or two are allowed. Our proof encodes a Turing rather than a two-register machine as in [14], and we believe some readers may find it simpler.

▶ **Theorem 14.** *The problems of checking whether $\Gamma, a \models \varphi$ and whether $\Gamma, a \models_{\mathit{fin}} \varphi$, given Γ, a, and φ are undecidable. This holds even when φ is of the form $r_1 \subseteq r_2$ and Γ contains only constraints of the following forms, where $\{r, r_1, r_2, r_3\} \subseteq \mathsf{N_R}$:*

$$r_1 \circ r_2 \subseteq r_3 \qquad r_1 \subseteq r_2 \circ r_3 \qquad [r](r_1 \circ r_2 \subseteq r_3) \qquad [r](r_1 \subseteq r_2 \circ r_3)$$

Proof. As in [14] we employ the notion of *conservative reduction classes* to simultaneously deal with general and finite implication. In addition, we see deciding $\Gamma, a \models \varphi_\mathcal{M}$ as checking unsatisfiability of the first order formula that corresponds to Γ and the negation of $\varphi_\mathcal{M}$. The same is true for finite implication. Let *FO* denote the set of FO formulae, X be a recursive subset of *FO*, and let $f : \mathit{FO} \to X$ be a recursive function such that:

- if $\beta \in \mathit{FO}$ is unsatisfiable, then $f(\beta)$ is unsatisfiable, and
- if $\beta \in \mathit{FO}$ has a finite model, then $f(\beta)$ has a finite model.

[4] In fact, the authors of [14] show that implication is r.e. complete, and finite implication co-r.e. complete.

Then X is a conservative reduction class and thus satisfiability of formulae in X is co-r.e.-complete and finite satisfiability r.e.-complete [12]. It is well known that for a first-order formula β we can build a procedure that takes no input and terminates iff β is unsatisfiable or has a finite model. Thus to show the undecidability of general and finite implication, it suffices to show how the computation of such a procedure can be simulated using path constraints. We consider a deterministic Turing Machine (TM) $\mathcal{M}$ with state set Q and alphabet Σ. We assume that $\mathcal{M}$ has two designated states q^1_{fin} and q^2_{fin}. We build Γ and $\varphi_{\mathcal{M}}$ such that the following conditions are satisfied:

1. If $\mathcal{M}$ reaches q^1_{fin} starting from the empty string as input, then $\Gamma_{\mathcal{M}}, a \models \varphi_{\mathcal{M}}$.
2. If $\mathcal{M}$ reaches q^2_{fin} starting from the empty string as input, then there exists a finite $\mathcal{I}$ such that $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ and $\mathcal{I}, a \not\models \varphi_{\mathcal{M}}$.

We assume that $\mathcal{M}$ starts at the initial tape position, and never moves to the left of it. Moreover, the tape is initially empty, that is, it only contains the blank symbol $\sqcup$. The transition function of $\mathcal{M}$ is of the form $\delta : Q \times \Sigma \to \Sigma \times Q \times \{\mathrm{R}, \mathrm{L}\}$, with the usual reading, where R and L stand for right and left, respectively. The initial state of M is q_{ini}.

The constraint $\varphi_{\mathcal{M}}$ takes the form $u_{\mathsf{ini}} \subseteq u_{\mathsf{halt}}$ where u_{ini} and u_{halt} are two role names. We define the set $\Gamma_{\mathcal{M}}$ next. Intuitively, the idea of the reduction is the following. Assume an arbitrary pointed structure $\mathcal{I}, a$ and let o denote $a^{\mathcal{I}}$. Whenever there is some $o' \in \Delta^{\mathcal{I}}$ such that $(o, o') \in u^{\mathcal{I}}_{\mathsf{ini}}$, the constraints in $\Gamma_{\mathcal{M}}$ will ensure that if $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$, then $\mathcal{I}$ contains a (possibly infinite) structure that represents the computation of $\mathcal{M}$, and that $(o, o') \in u^{\mathcal{I}}_{\mathsf{ini}}$ whenever the computation halts.

To provide an intuitive description of how this structure is enforced, we will use the term *r-arc* to refer to a pairs $(d, d') \in r^{\mathcal{I}}$ for a role name r. Now we describe the constraints in $\Gamma_{\mathcal{M}}$, which ensure that from the initial arc u_{ini} we build a full computation of $\mathcal{M}$. We use role names of the form $t_{q,\sigma}$ and $f_{q,\sigma}$ for each $q \in Q \cup \{\#\}$ and each $\sigma \in \Sigma \cup \{\sqcup\}$. Each of these symbols stands for a tape position containing the symbol σ. The marker $\#$ indicates that the head of $\mathcal{M}$ is not on the current position, while $q \in Q$ indicates that the head is on the current position and $\mathcal{M}$ is in state q. The symbols $f_{q,\sigma}$ are used to distinguish the right-most tape position, while regular 'inner' positions are represented by symbols $t_{q,\sigma}$. The first two constraints are prefix-empty and use the auxiliary role u_{aux}. They ensure that whenever there is an u_{ini} arc, there exists also an arc labeled $f_{q_{\mathsf{ini}},\sqcup}$, indicating that $\mathcal{M}$ is in state q_{ini}, the current tape position contains the symbol $\sqcup$, and the current tape position is the right-most one that has been visited so far:

$$u_{\mathsf{ini}} \subseteq u_{\mathsf{aux}} \circ u_{\mathsf{out}} \tag{1}$$

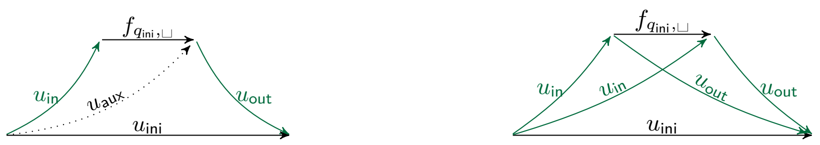
$$u_{\mathsf{aux}} \subseteq u_{\mathsf{in}} \circ f_{q_{\mathsf{ini}},\sqcup} \tag{2}$$

Note that there is an u_{in} arc from o to the beginning of $f_{q_{\mathsf{ini}},\sqcup}$, and an u_{out} arc from its end to o'. For the initial and final points of this $f_{q_{\mathsf{ini}},\sqcup}$ arc, and of all the arcs r that our construction will generate below in order to simulate the runs of $\mathcal{M}$, we want the role name u_{in} to connect o to the point, and u_{out} to connect the point to o'. We ensure this by adding the following constraints:

$$(u_{\mathsf{in}} \circ r \subseteq u_{\mathsf{in}}) \quad \text{for every } r \in \mathsf{N_R} \setminus \{u_{\mathsf{ini}}, u_{\mathsf{halt}}, u_{\mathsf{in}}, u_{\mathsf{out}}, u_{\mathsf{aux}},\} \text{ occurring in } \Gamma_{\mathcal{M}} \tag{3}$$

$$[u_{\mathsf{in}}](r \circ u_{\mathsf{out}} \subseteq u_{\mathsf{out}}) \quad \text{for every } r \in \mathsf{N_R} \setminus \{u_{\mathsf{ini}}, u_{\mathsf{halt}}, u_{\mathsf{in}}, u_{\mathsf{out}}, u_{\mathsf{aux}},\} \text{ occurring in } \Gamma_{\mathcal{M}} \tag{4}$$

Note that there is a pair of these constraints for every role name occurring in the rest of the proof, except for the u roles that do not have a direct correspondence with the configurations of $\mathcal{M}$. With these axioms, the initial and final points of the $f_{q_{\mathsf{ini}},\sqcup}$ arc are both reachable



Creating the arc $f_{q_{\mathsf{ini}},\sqcup}$ for the first tape position (constraints (1) and (2))

Connecting the new arc to the spy-points (constraints (3) and (4) for $r = f_{q_{\mathsf{ini}},\sqcup}$)

Figure 1 Model of $\Gamma_{\mathcal{M}}$ and $u_{\mathsf{ini}}(o, o')$ representing the initial configuration of $\mathcal{M}$.

from o, and both reach o'. The same holds for any other arc implied by the constraints below. The intended model of the constraints we have described so far is depicted in Figure 1.

Now we give the constraints that ensure that the computation of $\mathcal{M}$ is correctly simulated. The core idea is that (the initial and final points of the arc representing) each tape position will be linked via an n role (for $n$ext configuration) to (the initial and final points of an arc representing) the same tape position in the following configuration. Differently subindexed n roles and auxiliary '$d$iagonal' d roles are used to propagate information between configurations.

The first group handles the case where the machine is at the right-most visited tape position (that is, there is a symbol $f_{q\sigma}$), and executes a transition that moves to the right:

$$[u_{\mathsf{in}}](f_{q\sigma} \subseteq d_{\leftarrow\#\sigma'} \circ n_{fq'}) \qquad \text{for each } \delta(q,\sigma) = (q',\sigma',\mathrm{R}) \tag{5}$$

$$[u_{\mathsf{in}}](n_{fq} \subseteq f_{q_\sqcup} \circ d_f) \qquad \text{for each } q \in Q \tag{6}$$

After the tape contents have been updated at the current position, and the automaton has changed to the new state and moved right to a new final tape position, it is only left to go leftwards propagating to the next configuration the remaining tape contents. This is ensured by the following axioms:

$$[u_{\mathsf{in}}](t_{q\sigma} \circ n_{\leftarrow} \subseteq d_{\leftarrow q\sigma}) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{7}$$

$$[u_{\mathsf{in}}](d_{\leftarrow q\sigma} \subseteq n_{\leftarrow} \circ t_{q\sigma}) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{8}$$

The next group handles also the case where the machine is currently at the right-most visited tape position indicated by $f_{q\sigma}$, but this time it executes a transition that moves to the left:

$$[u_{\mathsf{in}}](f_{q\sigma} \subseteq d_{\leftarrow q' f \#\sigma'} \circ n_f) \qquad \text{for each } \delta(q,\sigma) = (q',\sigma',\mathrm{L}) \tag{9}$$

$$[u_{\mathsf{in}}](d_{\leftarrow q f \#\sigma} \subseteq n_{\leftarrow q} \circ f_{\#\sigma}) \qquad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\sqcup\} \tag{10}$$

$$[u_{\mathsf{in}}](t_{\#\sigma} \circ n_{\leftarrow q} \subseteq d_{\leftarrow q\sigma}) \qquad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\sqcup\} \tag{11}$$

We recall that constraints (7) and (8) already ensure that the remaining tape contents are properly propagated. Next we handle transitions to the right, from a non-final tape position:

$$[u_{\mathsf{in}}](t_{q\sigma} \subseteq d_{\leftarrow\#\sigma'} \circ n_{q'\rightarrow}) \qquad \text{for each } \delta(q,\sigma) = (q',\sigma',\mathrm{R}) \tag{12}$$

$$[u_{\mathsf{in}}](n_{q\rightarrow} \circ t_{\#\sigma} \subseteq d_{q\sigma\rightarrow}) \qquad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\sqcup\} \tag{13}$$

Additionally to the contents to the left, which have been taken care of, we also need to propagate tape contents to the right of the current position. We use constraints analogous to (7) and (8):

$$[u_{\mathsf{in}}](n_{\rightarrow} t_{q\sigma} \subseteq d_{q\sigma\rightarrow}) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{14}$$

$$[u_{\mathsf{in}}](d_{q\sigma\rightarrow} \subseteq t_{q\sigma} n_{\rightarrow}) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{15}$$

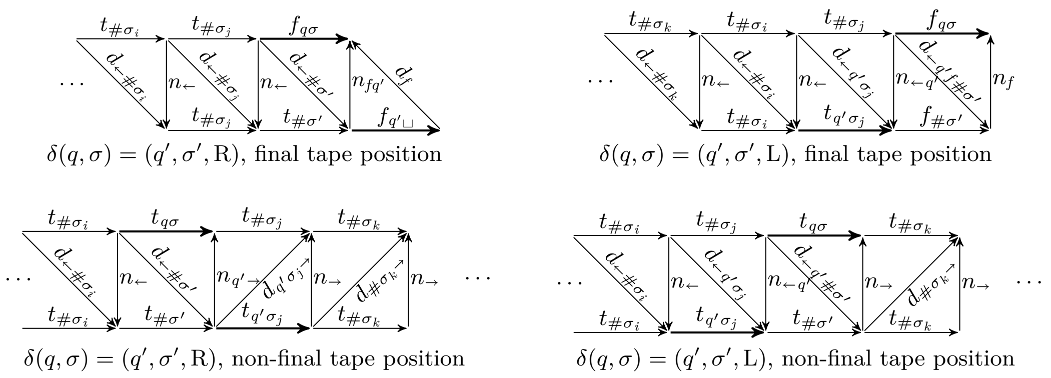

$\delta(q,\sigma) = (q',\sigma',\mathrm{R})$, final tape position
$\delta(q,\sigma) = (q',\sigma',\mathrm{L})$, final tape position
$\delta(q,\sigma) = (q',\sigma',\mathrm{R})$, non-final tape position
$\delta(q,\sigma) = (q',\sigma',\mathrm{L})$, non-final tape position

Figure 2 Parts of models of $\Gamma_{\mathcal{M}}$ and $u_{\mathsf{ini}}(o,o')$ representing the transitions of $\mathcal{M}$.

When we propagate to the right, we need to ensure that the final position $t_{q\sigma}$ is also copied to the next configuration:

$$[u_{\mathsf{in}}](n_{\rightarrow} \circ f_{q\sigma} \subseteq d_{q\sigma f}) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{16}$$

$$[u_{\mathsf{in}}](d_{q\sigma f} \subseteq f_{q\sigma} \circ n_f) \qquad \text{for each } q \in Q \cup \{\#\}, \sigma \in \Sigma \cup \{\sqcup\} \tag{17}$$

Finally we handle transitions to the left from a non-final tape position:

$$[u_{\mathsf{in}}](t_{q\sigma} \subseteq d_{\leftarrow q'\#\sigma'} \circ n_{\rightarrow}) \qquad \text{for each } \delta(q,\sigma) = (q',\sigma',\mathrm{L}) \tag{18}$$

$$[u_{\mathsf{in}}](d_{\leftarrow q\#\sigma} \subseteq n_{\leftarrow q} \circ t_{\#\sigma}) \qquad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\sqcup\} \tag{19}$$

$$[u_{\mathsf{in}}](t_{\#\sigma} \circ n_{\leftarrow q} \subseteq d_{\leftarrow q\sigma}) \qquad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\sqcup\} \tag{20}$$

Figure 2 illustrates how the constraints simulate the transitions of $\mathcal{M}$. For readability, the u_{in} arcs to and u_{out} arcs from every node are omitted.

An interpretation that is a model of the transitions we have presented correctly executes a computation of $\mathcal{M}$. It is only left to enforce the special role u_{halt} to hold between o and o' iff the computation halts in q^1_{fin}. We first ensure that if an arc $t_{q^1_{\mathsf{fin}},\sigma}$ or $f_{q^1_{\mathsf{fin}},\sigma}$ for some $\sigma \in \Sigma \cup \{\sqcup\}$ was generated (that is, q^1_{fin} was reached in the computation), an u_{halt} arc is created; note that if $\mathcal{M}$ halts in q^2_{fin} nothing enforces u_{halt} and there will be a model of the constraints where φ_M does not hold. Then we ensure that if there is an u_{halt} arc anywhere, then there is such an arc between o and o'. This is easy to do, exploiting the fact that all points are reachable via u_{ini} from o, and reach o' via u_{in}:

$$u_{\mathsf{in}} \circ t_{q^1_{\mathsf{fin}}\sigma} \subseteq u_{\mathsf{halt}} \qquad \text{for each } \sigma \in \Sigma \cup \{\sqcup\} \tag{21}$$

$$u_{\mathsf{in}} \circ f_{q^1_{\mathsf{fin}}\sigma} \subseteq u_{\mathsf{halt}} \qquad \text{for each } \sigma \in \Sigma \cup \{\sqcup\} \tag{22}$$

$$u_{\mathsf{halt}} \circ u_{\mathsf{out}} \subseteq u_{\mathsf{halt}} \tag{23}$$

Let $\Gamma_{\mathcal{M}}$ contain the constraints (1) – (23). We show that the reduction is as desired:

1. $\mathcal{M}$ reaches q^1_{fin} iff $\Gamma_{\mathcal{M}}, a \models \varphi_{\mathcal{M}}$.
   Suppose $\mathcal{M}$ reaches q^1_{fin}. Assume an arbitrary $\mathcal{I}, a$ such that $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$, and assume there is an arbitrary $d \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}}_{\mathsf{ini}}$. Since $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ and $\mathcal{M}$ reaches q^1_{fin}, then $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}}_{\mathsf{halt}}$ follows. This implies $\mathcal{I}, a \models \varphi_{\mathcal{M}}$ and hence $\Gamma, a \models \varphi_{\mathcal{M}}$. For the converse, assume $\Gamma, a \not\models \varphi_{\mathcal{M}}$. That is, there exists some $\mathcal{I}, a$ such that $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ but $\mathcal{I}, a \not\models \varphi_{\mathcal{M}}$. That is, there is $d \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}}_{\mathsf{ini}}$ but $(a^{\mathcal{I}}, d) \notin u^{\mathcal{I}}_{\mathsf{halt}}$. Since $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$, by constraints (3), (4), (21), and (23) we have $(a^{\mathcal{I}}, d) \notin u^{\mathcal{I}}_{\mathsf{halt}}$, and $\mathcal{I}$ cannot

contain any arcs of the form $t_{q^1_{\mathsf{fin}}\sigma}$ or $f_{q^1_{\mathsf{fin}}\sigma}$, which means that the computation of $\mathcal{M}$ does not reach the state q^1_{fin}.

2. If $\mathcal{M}$ reaches q^2_{fin}, then there exists a finite $\mathcal{I}$ such that $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ and $\mathcal{I}, a \not\models \varphi_{\mathcal{M}}$.
Suppose that there is a computation of $\mathcal{M}$ that reaches q^2_{fin}. Then this computation does not reach q^1_{fin}. Moreover, there is some $\mathcal{I}, a$ that simulates this computation, that is, such that $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}}_{\mathsf{ini}}$ and $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$. Since $\mathcal{M}$ does not reach q^1_{fin}, we can assume there are no arcs $t_{q^1_{\mathsf{fin}}\sigma}$ or $f_{q^1_{\mathsf{fin}}\sigma}$ in $\mathcal{I}$, and since the only constraints that imply u_{halt} are (21) and (23), we can assume $u^{\mathcal{I}}_{\mathsf{halt}} = \emptyset$. Hence we have $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ with $(a^{\mathcal{I}}, d) \in u^{\mathcal{I}}_{\mathsf{ini}}$ but $u^{\mathcal{I}}_{\mathsf{halt}} = \emptyset$ as desired.

This concludes the proof of Theorem 14. ◀

The same proof that we gave for the pointed semantics of path constraints applies also to the global semantics, even if we further restrict the constraints to be prefix-empty.

▶ **Theorem 15.** *The problems of deciding* $\Gamma \models \varphi$ *and* $\Gamma \models_{\mathit{fin}} \varphi$ *given* Γ *and* φ *are undecidable, even when* φ *is of the form* $r_1 \subseteq r_2$ *and* Γ *contains only constraints of the forms* $r_1 \circ r_2 \subseteq r_3$ *and* $r_1 \subseteq r_2 \circ r_3$*, for* $\{r_1, r_2, r_3\} \subseteq \mathsf{N_R}$.

Proof. It suffices to observe that in the proof of Theorem 14, all constrains $[R_p](R_\ell \subseteq R_r)$ with $R_p \neq \varepsilon$ have $R_p = u_{\mathsf{in}}$, and the only role of the prefix is to ensure that $R_\ell \subseteq R_r$ fires at all nodes, and not only at a. Under global semantics, this prefix is unnecessary and we can replace each constraint $[R_p](R_\ell \subseteq R_r)$ simply by $R_\ell \subseteq R_r$. ◀

We formulated Theorems 14 and 15 for path constraint implication, but we could just as well formulate them for the problem of deciding whether a fact $r_1(a,b)$ and a set of constraints Γ that are satisfied (at a with the pointed semantics for the prefixed version, or everywhere for the prefix-empty version) imply the existence of a pair of objects in r_2, which need not be a, b (note that in this case we don't need constraint (23)). Our negative results also apply to other languages that can express Γ. For instance, the prefix-empty version of Γ can be expressed in the following restricted class of *tuple generating dependencies*:

$$r_1(x,y), r_2(y,z) \to r_3(x,z) \qquad\qquad r_1(x,z) \to \exists y. r_2(x,y), r_3(y,z)$$

Hence we obtain a proof of undecidability of (finite) entailment of a query $\exists x, y.\, r'(x,y)$, from one single fact $r(a,b)$ in the presence of dependencies in this restricted class.

We note that the undecidability of path constraint implication shows the undecidability of (finite) satisfiability in $\mathcal{ZOI}^{\setminus}$. In fact, it shows the undecidability of (finite) satisfiability of a complex $\mathcal{ZOI}^{\setminus}$ role $id(\forall R.\bot) \circ (r_1 \setminus r_2)$, where R is a union of roles of the forms

$$(r_1 \circ r_2) \setminus r_3 \qquad r_1 \setminus (r_2 \circ r_3) \qquad r \circ ((r_1 \circ r_2) \setminus r_3) \qquad r \circ (r_1 \setminus (r_2 \circ r_3)) \tag{24}$$

Undecidability also applies to the (finite) entailment of $r_2(a,b)$ from $(id(\forall R.\bot) \circ r_1)(a,b)$. If we use a set $\mathcal{K}$ of inclusions of the form $\top \sqsubseteq \forall R_\ell \setminus R_r.\bot$ (where each R_ℓ and R_r is the concatenation of at most two role names) to enforce prefix-empty constraints to hold everywhere, we get undecidability of testing whether the KB $\mathcal{K} \wedge \{r_1(a,b)\}$ (finitely) entails $r_2(a,b)$.

We remark that the constraints in the proof above can also be written in $\mathsf{GXPath}_{\mathit{reg}}$, the extension of $\mathsf{GXPath}^{\mathit{path\text{-}pos}}_{\mathit{reg}}$ that allows for negation (and hence, intersection and difference) of path formulas. Hence Theorem 14 implies the undecidability of the (finite) implication of two $\mathsf{GXPath}_{\mathit{reg}}$ formulas ψ_1 and ψ_2, even when ψ_2 is restricted to a role name r_2 and ψ_1 is a formula of the form $[\neg\langle\xi\rangle] \circ r_1$ for r_1 a role name and ξ a union of roles of the forms in (24) above. Note that this result is tighter that the one in [27], whose proof heavily uses ε.

Improving the Upper bound for Prefix-empty Constraints. We have seen that under global semantics, or with non-empty prefixes, there is no hope for decidability of path constraint implication. However, prefix-empty constraints under the pointed semantics are expressible in plain $\mathcal{ZOI}$ without using role difference, using a nominal to denote the point at which the constraint is evaluated. Hence we can reduce their (finite) implication problem to (finite) KB satisfiability in $\mathcal{ZOI}$. This implies decidability and an ExpTime upper bound, thus significantly improving the previous 2ExpSpace bound shown in [3] for the subclass of prefix-empty one-way regular path constraints.

▶ **Theorem 16.** *Let Γ be a set of prefix-empty path constraints, φ a prefix-empty path constraint, and $a \in \mathsf{N_I}$. Then we can obtain a set of $\mathcal{ZOI}$ of axioms $\mathcal{T}_\Gamma$ and a concept C_φ such that $\Gamma \models \varphi$ iff $\{\mathcal{T}_\Gamma\} \wedge \neg C_\varphi$ is unsatisfiable. Moreover, $\mathcal{T}_\Gamma$ and C_φ can be constructed in linear time and the size of $\mathcal{T}_\Gamma$ and C_φ are both linear, in the combined sizes of Γ and φ.*

Proof. For each $\gamma = R_\ell \subseteq R_r \in \Gamma \cup \varphi$, let $\mathcal{T}_\gamma$ be the axiom $\{a\} \sqsubseteq \forall R_\ell.\exists \mathsf{inv}(R_r).\{a\}$ where $\mathsf{inv}(R_r)$ is the *inverted* R_r defined as follows, where $r \in \mathsf{N_R}$ and R, R' denote arbitrary roles:

$$\begin{array}{ccc}
\mathsf{inv}(r) = r^- & \mathsf{inv}(r^-) = r & \mathsf{inv}(\{(a,b)\}) = \{(b,a)\} \\
\mathsf{inv}(id(C)) = id(C) & \mathsf{inv}(R \cap R') = \mathsf{inv}(R) \cap \mathsf{inv}(R') & \mathsf{inv}(R \cup R') = \mathsf{inv}(R) \cup \mathsf{inv}(R') \\
\mathsf{inv}(R^*) = (\mathsf{inv}(R))^* & \mathsf{inv}(R \setminus R') = \mathsf{inv}(R) \setminus \mathsf{inv}(R') & \mathsf{inv}(R \circ R') = \mathsf{inv}(R') \circ \mathsf{inv}(R)
\end{array}$$

and $\mathsf{inv}(\varepsilon) = \varepsilon$. It is easy to see that $(d, d') \in R^\mathcal{I}$ iff $(d', d) \in \mathsf{inv}(R)^\mathcal{I}$ for every $d, d' \in \Delta^\mathcal{I}$. It is then a straightforward consequence of the semantics of $\mathcal{ZOI}$ and of γ, that for every pointed instance $\mathcal{I}, a$ we have $\mathcal{I}, a \models \gamma$ if and only if $\mathcal{I} \models \mathcal{T}_\gamma$. Hence it easily follows that $\Gamma \models \varphi$ if and only if $(\bigcup_{\gamma\in\Gamma} \mathcal{T}_\gamma) \wedge \neg\mathcal{T}_\varphi$ is unsatisfiable. ◀

From this reduction and Theorem 5 we get:

▶ **Corollary 17.** *The unrestricted and the finite pointed implication problems for prefix-empty path constraints are decidable in* ExpTime.

We can combine this result and Corollary 12, obtaining that entailment is decidable in ExpTime whenever all constraints are prefix-empty or involve only simple roles. Although this is a strong restriction, it still allows for fairly non-trivial constraints. Apart from capturing these relevant decidable subclasses of path constraints, $\mathcal{ZOI}$ can also express many other natural constraints, some of which are not easily expressible as path constraints. For example, ϕ_5 in Example 3 does not directly correspond to a path constraint. $\mathcal{ZOI}$ provides flexible means to express quite involved expressions, for example, that a course required in an undergraduate program must be taught by a faculty member that is a member of an institute, or a suborganization of it:

$$\mathsf{Course} \sqcap \exists \mathsf{requires}^-.\mathsf{UndergradProg} \sqsubseteq \exists \mathsf{teaches}^-.\big(\exists(\mathsf{memberOf} \circ \mathsf{partOf}^*).\mathsf{Inst}\big).$$

To conclude this section, we remark that both the pointed and the global semantics are directly supported within $\mathcal{ZOI}$: the logic has a global semantics, but we can use nominals to ensure that any assertion or inclusion only 'fires' at the interpretation of a given individual.

4 Verification of Evolving Graph Structured Data

We have advocated the use of $\mathcal{ZOI}$ as a constraint language for GSD. We define a language for manipulating GSD and show that it is possible to effectively reason about the preservation of $\mathcal{ZOI}$ constraints when data instances evolve as a result of operations in this language.

Updating Graph Structured Data. The basic operations in our manipulation language allow 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 in $\mathcal{ZOI}$. The language allows, in particular, to select one object and add it to or remove it from a concept name, using a nominal $\{a\}$. It similarly allows to add or remove pairs of objects to/from role names using a nominal role $\{(a,b)\}$. Moreover, we can add to or delete from some role or concept the answers to query expressed as a complex concept or role (in the latter case, a so-called *regular path query*). We also allow for variables in the place of individuals, to have a more natural manipulation language with parameters that can be instantiated with different values. Finally, these basic actions can be combined into complex ones using composition and conditional actions.

The language we define next is like the one in [4], but there basic actions use concepts and roles in a different DL called $\mathcal{ALCHOIQ}br$, instead of $\mathcal{ZOI}$. $\mathcal{ALCHOIQ}br$ does not allow for regular expressions as roles, hence it cannot express path queries and path constraints. On the other hand, it supports *number restrictions*, which are not allowed in $\mathcal{ZOI}$. We note that the language in [4] allows roles of the form $R|_C$ (or $\mathsf{inv}(R)|_C$), which stand for the pairs of objects in R whose first (resp., second) component is an instance of C. Such roles are expressible in $\mathcal{ZOI}$ using $R \circ id(C)$.

▶ **Definition 18** (Action language). In what follows, additionally to $\mathsf{N_I}$, $\mathsf{N_C}$ and $\mathsf{N_R}$, we consider a countably infinite set $\mathsf{N_V}$ of variables, disjoint from the other sets. We use $\mathcal{ZOI}_\mathsf{V}$ concepts, roles, and KBs, which are defined as for $\mathcal{ZOI}$, but allowing for variables in the place of individuals, that is, atomic concepts take the forms A and $\{t\}$, and atomic roles the forms r, r^- and $\{(t,t')\}$, where $t,t' \in \mathsf{N_V} \cup \mathsf{N_I}$, $A \in \mathsf{N_C}$, $r \in \mathsf{N_R} \setminus \{\mathsf{T}\}$.

Basics action β and *(complex) actions* α are defined by the following grammar:

$$\beta \longrightarrow (A \oplus C) \mid (A \ominus C) \mid (r \oplus R) \mid (r \ominus R) \qquad \alpha \longrightarrow \epsilon \mid \beta \cdot \alpha \mid (\mathcal{K}\,?\,\alpha[\![\alpha]\!]) \cdot \alpha$$

with A a concept name, C an arbitrary $\mathcal{ZOI}_\mathsf{V}$ concept, r a role name, R an arbitrary $\mathcal{ZOI}_\mathsf{V}$ role, and $\mathcal{K}$ an arbitrary $\mathcal{ZOI}_\mathsf{V}$ KB. The symbol ϵ denotes the *empty action.*

We call a concept, role, KB or an action *ground* if it has no variables. A *substitution* is a function σ from $\mathsf{N_V}$ to $\mathsf{N_I}$. For a concept, role, KB or an action 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 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 instance $\mathcal{I}$ is the addition of the content of $C^\mathcal{I}$ to $A^\mathcal{I}$. Similarly, $(A \ominus C)$ removes the content of $C^\mathcal{I}$ from $A^\mathcal{I}$. The two operations can also be performed on 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 instance is a model of $\mathcal{K}$, and α_2 is executed otherwise. If $\alpha_2 = \epsilon$ 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 introduce the notion of *instance update.*

▶ **Definition 19** (Instance update, semantics of actions). Assume an instance $\mathcal{I}$ 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 instance $\mathcal{I}'$ such that $\Delta^{\mathcal{I}'} = \Delta^\mathcal{I}$, $E^{\mathcal{I}'} = E^\mathcal{I} \cup W$ (resp., $E^{\mathcal{I}'} = E^\mathcal{I} \setminus W$), and $E_1^{\mathcal{I}'} = E_1^\mathcal{I}$, for all symbols $E_1 \neq E$. Given a ground action α, we define a mapping S_α from instances to instances as follows:

$$S_\epsilon(\mathcal{I}) = \mathcal{I} \qquad S_{(A \oplus C)\cdot\alpha}(\mathcal{I}) = S_\alpha(\mathcal{I} \oplus_A C^{\mathcal{I}})$$

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

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 20.** Consider the next action with a free variable x, and its ground instance α_{dl}:

$$\begin{aligned}\text{RemoveCourse}(x) = {} & \big(\mathsf{Course} \ominus \{x\}\big) \cdot \big(\mathsf{requires} \ominus (\mathsf{requires} \circ id(\{x\}))\big) \\ & \cdot \big(\mathsf{offers} \ominus (\mathsf{offers} \circ id(\{x\}))\big) \cdot \big(\mathsf{partOf} \ominus (id(\{x\}) \circ \mathsf{partOf})\big)\end{aligned}$$

$$\begin{aligned}\alpha_{dl} = {} & \big(\mathsf{Course} \ominus \{\text{DLs:CS451}\}\big) \cdot \big(\mathsf{requires} \ominus (\mathsf{requires} \circ id(\{\text{DLs:CS451}\}))\big) \\ & \cdot \big(\mathsf{offers} \ominus (\mathsf{offers} \circ id(\{\text{DLs:CS451}\}))\big) \cdot \big(\mathsf{partOf} \ominus (id(\{\text{DLs:CS451}\}) \circ \mathsf{partOf})\big)\end{aligned}$$

In $S_{\alpha_{dl}}(\mathcal{I}_{\mathsf{Uni}})$ we have the following changes, and the rest of the instance remains unchanged:

$$\begin{aligned} \mathsf{Course}^{\mathcal{I}} &= \{\text{DataStruct:CS202}, \text{FoundDBs:CS327}\} \\ \mathsf{requires}^{\mathcal{I}} &= \{(\text{BSc\_CSci}, \text{DataStruct:CS202}), (\text{MSc\_CompLogic}, \text{FoundDBs:CS327}), \\ &\quad (\text{MSc\_CompLogic}, \text{mod\_KR})\} \\ \mathsf{offers}^{\mathcal{I}} &= \{(\text{CS\_Dept}, \text{BSc\_CSci}), (\text{CS\_Dept}, \text{MSc\_CompLogic}), \\ &\quad (\text{CS\_Dept}, \text{MSc\_Bioinformatics})\{(\text{CS\_Dept}, \text{DataStruct:CS202}), \\ &\quad (\text{CS\_Dept}, \text{FoundDBs:CS327})\} \\ \mathsf{partOf}^{\mathcal{I}} &= \{\} \end{aligned}$$

The Static Verification Problem. We consider now the scenario where DL KBs are used to impose integrity constraints on GSD. A basic reasoning problem for analyzing the effects of actions in the presence of integrity constraints 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 $\mathcal{K}$.

▶ **Definition 21** (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* consists on deciding, given an action α and a KB $\mathcal{K}$, whether α is $\mathcal{K}$-preserving. ◀

▶ **Example 22.** Recall the constraints $\mathcal{K}_{\mathsf{Uni}}$ from Ex. 3, and the action α_{dl} from Ex. 20. Note that α_{dl} is not $\mathcal{K}_{\mathsf{Uni}}$-preserving. In fact, this is witnessed by out instance $\mathcal{I}_{\mathsf{Uni}}$. We saw that in $S_{\alpha_{dl}}(\mathcal{I}_{\mathsf{Uni}})$ we have $(\text{MSc\_CompLogic}, \text{mod\_KR}) \in \mathsf{requires}^{\mathcal{I}}$, but $\text{mod\_KR} \notin (\exists \mathsf{partOf}^{-*}.\mathsf{Course})^{\mathcal{I}}$, that is, the mandatory KR module does not contain any courses, violating $\phi_4 = \top \sqsubseteq \forall \mathsf{requires}.(\exists \mathsf{partOf}^{-*}.\mathsf{Course})$.

Our technique for static verification relies on a transformation $\mathsf{TR}_\alpha(\mathcal{K})$ that rewrites $\mathcal{K}$ incorporating the effects of an action α. The technique is similar in spirit to *regression* in reasoning about actions [29], and it can be seen as a way to compute the *weakest precondition* of α and $\mathcal{K}$. 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.

▶ **Definition 23.** Given a $\mathcal{ZOI}$ 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 a ground action α, we define $\mathsf{TR}_\alpha(\mathcal{K})$ as follows.

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

Assume a ground action α and a $\mathcal{ZOI}$ KB $\mathcal{K}$. Note that the transformation $\mathsf{TR}_\alpha(\mathcal{K})$ may introduce role differences involving complex roles from α. Hence $\mathsf{TR}_\alpha(\mathcal{K})$ is a $\mathcal{ZOI}^\setminus$ and need not be a $\mathcal{ZOI}$ KB. Note also that the size of $\mathsf{TR}_\alpha(\mathcal{K})$ might be exponential in the size of α. By employing the same argument as [4], we see that the transformation correctly captures the effects of complex actions. In particular, for every interpretation $\mathcal{I}$, we have $S_\alpha(\mathcal{I})\models\mathcal{K}$ iff $\mathcal{I}\models\mathsf{TR}_\alpha(\mathcal{K})$. With the transformation $\mathsf{TR}_\alpha(\mathcal{K})$ above we have a reduction from static verification to finite (un)satisfiability of $\mathcal{ZOI}^\setminus$ 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 24** ([4]). *For a (complex) action α and a KB $\mathcal{K}$, 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 $\mathcal{K}$.*

Undecidability of unrestricted static verification. The first and foremost consequence of this reduction is that for the action language we have defined, the static verification problem is undecidable, even if the input $\mathcal{K}$ is trivial, and the actions are quite restricted:

▶ **Theorem 25.** *Deciding whether α is $\mathcal{K}$-preserving is undecidable, even when $\mathcal{K}$ is a trivial KB of the form $r(a,b)$, and α is just a sequence of basic actions of the forms $(r\oplus R)$ and $(r\ominus R)$, with R a sequence of one or two concatenated role names.*

Proof. We provide a reduction from deciding $\Gamma\models_{\mathit{fin}} r_1\subseteq r_2$, where Γ contains only constraints of the forms $r_1\circ r_2\subseteq r_3$ and $r_1\subseteq r_2\circ r_3$ for $\{r_1,r_2,r_3\}\subseteq\mathsf{N_R}$. We have seen above that this problem is undecidable. In particular, we construct an action α such that $\Gamma\models_{\mathit{fin}} r_1\subseteq r_2$ iff α is $r_1(a,b)$-preserving. Let $R_1^1\subseteq R_2^1,\dots,R_1^n\subseteq R_2^n$ be an enumeration of all constraints in Γ. For every $1\le i\le n$, let p_1^i and p_2^i be fresh role names. Then α is the concatenation of the following actions in the given order:

- $(r_1\ominus r_1)\cdot(r_1\oplus r_2)$
- $(p_1^1\ominus p_1^1)\cdot\dots\cdot(p_1^n\ominus p_1^n)\cdot(p_2^1\ominus p_2^1)\cdot\dots\cdot(p_2^n\ominus p_2^n)$
- $(p_1^1\oplus R_1^1)\cdot\dots\cdot(p_1^n\oplus R_1^n)\cdot(p_2^1\oplus R_2^1)\cdot\dots\cdot(p_2^n\oplus R_2^n)$
- $(p_1^1\ominus p_2^1)\cdot\dots\cdot(p_1^n\ominus p_2^n)\cdot(r_1\oplus p_1^1)\cdot\dots\cdot(r_1\oplus p_1^n)$.

Recall that α is not $r_1(a,b)$-preserving iff $r_1(a,b)\wedge\neg\mathsf{TR}_\alpha(r_1(a,b))$ is finitely satisfiable. It's not hard to see that $\mathsf{TR}_\alpha(r_1(a,b))=R_\Gamma(a,b)$, where R_Γ is equivalent to the role $r_2\cup(R_1^1\setminus R_2^1)\cup\dots\cup(R_1^n\setminus R_2^n)$. Thus $r_1(a,b)\wedge\neg\mathsf{TR}_\alpha(r_1(a,b))$ is finitely satisfiable iff $\Gamma\not\models_{\mathit{fin}} r_1\subseteq r_2$. ◀

Unfortunately we cannot allow for complex roles in our actions, not even of the form $r\circ r'$, but we get positive results if we restrict actions to *simple* roles.

An undesired effect of disallowing complex roles in actions is that we cannot express $r|_C$ as $r\circ id(C)$, and as our examples illustrate, this construct is quite useful. For nominals we can, however, simulate $r|_{\{a\}}$ in $\mathcal{ZOI}$. We use a special role name $\mathsf{T}|_{\{a\}}$ with the intended semantics $(\mathsf{T}|_{\{a\}})^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \{a\}^{\mathcal{I}}$, and write the simple role $r \cap \mathsf{T}|_{\{a\}}$ in the place of $r|_{\{a\}}$. The intended meaning of $\mathsf{T}|_{\{a\}}$ is easily enforced by adding $\top \sqsubseteq \exists \mathsf{T}|_{\{a\}}.a$ to any $\mathcal{ZOI}$ KB.

We call an action α *role-restricted* if in every basic action of the form $(r \oplus R)$ or $(r \ominus R)$, we have that R is a simple $\mathcal{ZOI}_{\mathsf{V}}$ role that may use the special role names $\mathsf{T}|_{\{a\}}$.

Note α_{dl} in Example 20 can be rewritten as a role-restricted action as follows:

$$\alpha'_{dl} = \big(\mathsf{Course} \ominus \{\mathsf{DLs{:}CS451}\}\big) \cdot \big(\mathsf{requires} \ominus (\mathsf{requires} \cap \mathsf{T}|_{\{\mathsf{DLs:CS451}\}})\big) \cdot \\ \big(\mathsf{offers} \ominus (\mathsf{offers} \cap \mathsf{T}|_{\{\mathsf{DLs:CS451}\}})\big) \cdot \big(\mathsf{partOf} \ominus (\mathsf{partOf} \cap (\mathsf{T}|^{-}_{\{\mathsf{DLs:CS451}\}})\big).$$

▶ **Theorem 26.** *Deciding whether α is $\mathcal{K}$-preserving for a given $\mathcal{ZOI}$ KB $\mathcal{K}$ and a role-restricted α is* ExpTime*-complete.*

Proof sketch. Since the union and the difference of simple roles are simple roles, it is not hard to see that the result of iteratively replacing role names by simple roles involving union and difference in a $\mathcal{ZOI}$ role results in a $\mathcal{ZOI}$ role. Hence, for any $\mathcal{ZOI}$ KB $\mathcal{K}$ and a role-restricted action α, the KB $\mathsf{TR}_\alpha(\mathcal{K})$ is not only a $\mathcal{ZOI}^{\setminus}$ KB but also a $\mathcal{ZOI}$ KB (i.e., difference is applied to simple roles only). Then from the decidability of (finite) satisfiability of $\mathcal{ZOI}$ it follows that checking whether α is $\mathcal{K}$-preserving is decidable. For the complexity upper bound, recall that the size of $\mathsf{TR}_\alpha(\mathcal{K})$ might be exponential in the size of α. However, as argued in [4], there are only exponentially many conjunctive clauses in disjunctive normal from of $\mathcal{K} \wedge \dot{\neg}\mathsf{TR}_{\alpha^*}(\mathcal{K})$, each with size polynomial in the size of α and $\mathcal{K}$. Thus from Theorems 5 and 24 we obtain the desired result. ◀

5 Conclusions and Outlook

The main goal of this work was to advocate the use of the DL $\mathcal{ZOI}$ to specify constraints over graph structured data and to show the decidability of static verification in a rich language for manipulating such data. Along the way, we have shown several undecidability and complexity results that concern not only our setting, but also formalisms that were introduced in the 90s, as well as recently introduced query languages for GSD like GXPath_{reg}. In our future work we aim at providing some support for identification constraints, which is clearly desirable but naturally requires equality reasoning. This is challenging, as e.g., the decidability of the extension of $\mathcal{ZOI}$ where some roles must be interpreted as partial functions is a long standing open problem.

Acknowledgements. This work was supported by the EU IP project Optique (*Scalable End-user Access to Big Data*), grant agreement n. FP7-318338, the Vienna Science and Technology Fund project ICT12-15, and the Austrian Science Fund projects P25207 and T515.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

2 Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

3 Serge Abiteboul and Victor Vianu. Regular path queries with constraints. *J. of Computer and System Sciences*, 58(3):428–452, 1999.

4 Shqiponja Ahmetaj, Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Managing change in Graph-structured Data using Description Logics. In *Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 966–973. AAAI Press, 2014.

5 Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys*, 40(1):1:1–1:39, 2008. `doi:10.1145/1322432.1322433`.

6 Marcelo Arenas, Wenfei Fan, and Leonid Libkin. On the complexity of verifying consistency of XML specifications. *SIAM J. on Computing*, 38(3):841–880, 2008. `doi:10.1137/050646895`.

7 Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. Reasoning over extended ER models. In *Proc. of the 26th Int. Conf. on Conceptual Modeling (ER)*, volume 4801 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2007.

8 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

9 Jie Bao et al. OWL 2 Web Ontology Language document overview (second edition). W3C Recommendation, World Wide Web Consortium, December 2012. URL: `http://www.w3.org/TR/owl2-overview/`.

10 Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.

11 Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web. Web Logic Rules – 11th Int. Summer School Tutorial Lectures (RW)*, volume 9203 of *Lecture Notes in Computer Science*, pages 218–307. Springer, 2015. `doi:10.1007/978-3-319-21768-0_9`.

12 Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer, 1997.

13 Peter Buneman. Semistructured data. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 117–121, 1997.

14 Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured databases. *J. of Computer and System Sciences*, 61(2):146–193, 2000. `doi:10.1006/jcss.2000.1710`.

15 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive query containment and answering under description logics constraints. *ACM Trans. on Computational Logic*, 9(3):22.1–22.31, 2008.

16 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Reasoning on regular path queries. *SIGMOD Record*, 32(4):83–92, 2003.

17 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. *Theoretical Computer Science*, 371(3):169–182, 2007.

18 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 714–720, 2009.

19 Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Evolution of *DL-Lite* knowledge bases. In *Proc. of the 9th Int. Semantic Web Conf. (ISWC)*, volume 6496 of *Lecture Notes in Computer Science*, pages 112–128. Springer, 2010.

20 Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Containment of regular path queries under description logic constraints. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 805–812, 2011.

21 Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms.* PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

22 Alin Deutsch and Val Tannen. Optimization properties for classes of conjunctive regular path queries. In *Proc. of the 8th Int. Workshop on Database Programming Languages (DBPL)*, volume 2397 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2001.

23 Wenfei Fan and Leonid Libkin. On XML integrity constraints in the presence of DTDs. *J. of the ACM*, 49(3):368–406, 2002.

24 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.

25 Erich Grädel and Martin Otto. On logics with two variables. *Theoretical Computer Science*, 224:73–113, 1999.

26 Gösta Grahne and Alex Thomo. Query containment and rewriting using views for regular path queries under constraints. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 111–122, 2003.

27 Egor V. Kostylev, Juan L. Reutter, and Domagoj Vrgoc. Containment of data graph queries. In *Proc. of the 17th Int. Conf. on Database Theory (ICDT)*, pages 131–142. OpenProceedings.org, 2014. `doi:10.5441/002/icdt.2014.16`.

28 Maurizio Lenzerini. Ontology-based data management. In *Proc. of the 20th Int. Conf. on Information and Knowledge Management (CIKM)*, pages 5–6, 2011. `doi:10.1145/2063576.2063582`.

29 H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.

30 Leonid Libkin, Wim Martens, and Domagoj Vrgoc. Querying graph databases with XPath. In *Proc. of the 16th Int. Conf. on Database Theory (ICDT)*, pages 129–140. ACMP, 2013. `doi:10.1145/2448496.2448513`.

31 Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Updating description logic ABoxes. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pages 46–56, 2006.

32 Michael Mortimer. On languages with two variables. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:135–140, 1975.

33 Magdalena Ortiz. Ontology based query answering: The story so far. In *Proc. of the 7th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW)*, volume 1087 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2013.

34 Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008. `doi:10.1007/978-3-540-77688-8_5`.

35 Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 466–471, 1991.

36 Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, 2012. `doi:10.1145/2206869.2206879`.