Verification of Evolving Graph-structured Data under Expressive Path Constraints

Diego Calvanese¹, Magdalena Ortiz², and Mantas Šimkus²

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

Abstract

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. 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 also 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.2016.1

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 the ability to express paths that return to the initial point (i.e., a form of loop) [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 heterogenous data sources, in order to facilitate their management and provide improved access to them by leveraging domain knowledge. As for plain databases, also in this setting, query answering for (unions of) conjunctive queries is tightly related to query containment, and has been studied extensively for variants of both lightweight [11] and expressive DLs [15, 33].

In this paper we show that DLs are adequate also as constraint languages for GSD. We focus on the expressive DL ZOI, also known as ALCOI 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.¹ 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) 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 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 ZOI enjoys finite model property (FMP), as we are able to prove. This is a crucial stepping stone for the results in this paper, and an interesting contribution on its own right. Indeed, while the

¹ We note that when regular expressions are allowed as query atoms, query containment under DL constraints in general cannot anymore be reduced to query answering, and requires ad-hoc techniques [20].
FMP has been long known for the closely related converse PDL [24, 21]. 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. The undecidability of general path constraint implication implies that static verification becomes undecidable when complex roles are allowed in actions. However, we show that 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 ZOI constraints to satisfiability of ZOI KBs, obtaining a tight E\(\text{XP}\)T\(\text{IME}\) upper bound for the former.

The paper is organized as follows. In Section 2 we introduce the DL ZOI. Our first technical contribution is to establish the finite model property of ZOI, necessary to employ known complexity bounds for (unrestricted) reasoning in ZOI to the setting of reasoning about (finite instances of) GSD. Section 3 is devoted to path constraint languages, and the relation between traditional such languages and (variations of) ZOI. The main technical contribution of this section is to tighten the undecidability of path constraint implication shown in [14] to the case where neither inverse relations nor equality are allowed. We also generalize the decidability of the restricted class in [3] to a richer class of 2-way constraints, and improve the upper bound from 2ExpSpace to ExpTime, by reducing the problem to reasoning in ZOI. Section 4 studies the static verification of ZOI constraints over GSDs, for actions expressed in the language proposed in [4]. We show that the problem is undecidable if arbitrary ZOI roles occur in actions, and impose suitable restrictions to obtain ExpTime decidability using the techniques of [4]. We conclude the paper with Section 5.

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 ZOI, also known as ALCOI\textsubscript{reg} [18], to express constraints over graph structured data. This is natural as, like other DLs, 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 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 (ZOI syntax). We consider fixed, countably infinite sets \(N_C\) of concept names, \(N_R\) of role names, and \(N_I\) of individual names. We assume that the set \(N_C\) contains the special concepts \(\top\) (top) and \(\bot\) (bottom), while \(N_R\) contains the top (universal) role \(T\) and the bottom (empty) role \(B\). We define (ZOI) atomic concepts \(B\), concepts \(C, C'\), atomic roles \(P\), simple roles \(S, S'\), and roles \(R, R'\), where \(a, b \in N_I, A \in N_C, r \in N_R, r \neq T\), according to the following syntax:

---

2 We note that, although we consider a setting where updates and constraints are expressed in a DL, our updates are assumed to be performed on complete data, and the constraints are not used to infer new knowledge before or after the update. Thus, our setting differs from the one studied, e.g., in [31, 19], and we do not run in the expressiveness issues considered there.
Verification of Evolving GSD under Expressive Path Constraints

\[\begin{align*}
B & \rightarrow A \mid \{a\} \\
C, C' & \rightarrow B \mid \neg C \mid C \cap C' \mid C \cup C' \mid \forall R.C \mid \exists R.C \mid \exists S.Self \\
S, S' & \rightarrow P \mid S \cap S' \mid S \cup S' \mid S \setminus S' \\
R, R' & \rightarrow T \mid \varepsilon \mid id(C) \mid S \mid R \cup R' \mid R \circ R' \mid R''
\end{align*}\]

A ZOI expression is a concept or a role. Subconcepts, subroles, and subexpressions are defined in the natural way. An expression \(\{a\}\) is called a nominal concept, and \(\{(a, b)\}\) a nominal role.

To ease presentation we have included the role \(\varepsilon\), which, as we shall see, has the same meaning as \(id(T)\) and captures the identity relation. We have also included a non-standard role construct \(\{(a, b)\}\), which can be easily simulated using standard ZOI constructs, as we show later.

We use 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 (ZOI knowledge bases).** A concept inclusion is an expression of the form \(C \subseteq C'\), where \(C, C'\) are arbitrary concepts, and a role inclusion is an expression of the form \(S \subseteq S'\), 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 \(\{a, a'\} \subseteq N_t\). Then, (ZOI) knowledge bases (KBs) are defined inductively as follows: (i) every inclusion and every assertion is a KB; (ii) if \(K, K'\) are KBs, so are \(K \wedge K', K \lor K', \neg K\).

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

\[\begin{align*}
\{a\}^I &= \{a^I\} \\
\neg C^I &= \Delta^I \setminus C^I \\
(C \cap C')^I &= C^I \cap C'^I \\
(C \cup C')^I &= C^I \cup C'^I \\
\forall R.C^I &= \{x \mid \forall y.(x, y) \in R^I \rightarrow y \in C^I\} \\
\exists R.C^I &= \{x \mid \exists y.(x, y) \in R^I \land y \in C^I\} \\
\exists S.Self^I &= \{x \mid (x, x) \in S^I\}
\end{align*}\]

\[\begin{align*}
\{(a, b)\}^I &= \{(a^I, b^I)\} \\
(r^-)^I &= \{(y, x) \mid (x, y) \in r^I\} \\
(S \setminus S')^I &= S^I \setminus S'^I \\
(S \cap S')^I &= S^I \cap S'^I \\
(R \cup R')^I &= R^I \cup R'^I \\
(R \circ R')^I &= R^I \circ R'^I \\
(id(C))^I &= \{x, x \mid x \in C^I\} \\
(e)^I &= \{(x, x) \mid x \in \Delta^I\}
\end{align*}\]

where \(\cap, \cup, \text{ and } \setminus\) are overloaded to denote also the standard set-theoretic operations, \(\circ\) to denote composition, and \(\rightarrow\) to denote the reflexive transitive closure of a binary relation.

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

**Remark (Simulating nominal roles).** The constructor \(\{(a, b)\}\) is just syntactic sugar. Indeed, it can be easily simulated in \(K\) by replacing each occurrence of \(\{(a, b)\}\) by a fresh role name \(r_{ab}\), and conjunctively adding to \(K\) the KB \(r_{ab}(a, b) \land (\exists r_{ab}.T \subseteq \{a\}) \land (T \subseteq \forall r_{ab}.\{b\})\). This ensures that \(r_{ab}^I = \{(a^I, b^I)\}\) in every model of the modified KB.
Example 3. As a running example, we consider the following self-explanatory instance $I_{\text{uni}}$.

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

Note that, for simplicity, in the example we interpret individuals as themselves (i.e., we make the standard name assumption). Consider the KB $K_{\text{uni}}$ defined as the conjunction of the following $ZOI$ constraints: $\phi_1$ says that the domain of ‘offers’ are the departments, and $\phi_2$ says that its range is the union of programs and courses. Similarly, $\phi_3$ and $\phi_4$ restrict the domain of ‘requires’ to programs, and its range to courses other entities that comprise courses, like modules. Finally $\phi_5$ says that every course that is required (directly, or because it is part of a required module) must be offered.

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

Note that all these constraints are satisfied by our instance, that is, $I_{\text{uni}} \models K_{\text{uni}}$.

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

As a basic reasoning task in $ZOI$ we consider KB satisfiability, which consists in deciding, given a KB $K$, whether $K$ 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.

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

Theorem 4. Given a $ZOI$ KB $K$, one can construct in linear time a $ZOI$ concept $C_K$ such that $K$ is satisfiable if and only if $C_K$ is so.

From this result and the EXPTIME upper bound for concept satisfiability given in [18], it follows immediately that satisfiability of $ZOI$ 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 $ZOI$ KBs is an EXPTIME-complete problem.

Finite Model Reasoning in $ZOI$. 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 ZOI has not been addressed so far. However, as we show in the following, 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 $K$ 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 $K$ be a ZOI KB. Then $K$ is satisfiable if and only if $K$ is finitely satisfiable.

### 3 Path Constraints

We define a language for path constraints inspired by [3, 14, 26]; we will see below that it is closely related to the DL ZOI.

**Definition 7 (Path constraints).** A path constraint $\varphi$ has the form $[R_p](R_\ell \subseteq R_r)$, where $R_p$, $R_\ell$, and $R_r$ are arbitrary ZOI roles. The role $R_p$ is called prefix of $\varphi$, while the roles $R_\ell$ and $R_r$ are respectively called the left tail and the right tail of $\varphi$. If $R_p = \varepsilon$, we call $\varphi$ a prefix-empty constraint\(^3\), 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 $N_R \cup \{\varepsilon\}$ using $\circ$, $\cup$, and $\ast$ is called a (one-way) regular path role, and if additionally it does not contain $\cup$ or $\ast$ 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 $R_\ell$ and $R_r$ are one-way regular path roles. If, in addition, $R_\ell$ 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, $r_1^{-1} \circ \cdots \circ r_n^{-1}$ with $n \geq 0$).\(^4\)

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 call this the pointed semantics, and 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 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 $I$, we define $\varphi^I$ as the set of objects $d \in \Delta^I$ that satisfy the following condition: for each $d', d'' \in \Delta^I$, if $(d, d') \in R_p^I$ and $(d', d'') \in R_r^I$, then $(d', d'') \in R_r^I$.

---

\(^3\) Prefix-empty constraints were called simple in [14]. We use a different name to avoid confusion with the simple roles of Definition 1.

\(^4\) We note that [14] uses a different syntax with explicit variables.
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 $\varphi$, and write $\mathcal{I}, a \models \varphi$ if $a \in \mathcal{I}^2$. Similarly, we write $\mathcal{I}, a \models \Gamma$ for a set $\Gamma$ 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 $\Gamma$, and write $\Gamma, a \models_{\text{fin}} \varphi$ if $\mathcal{I}, a \models \varphi$ for every finite pointed model $\mathcal{I}, a$ of $\Gamma$. The (finite) pointed implication problem consists in deciding, given an individual $a$, a set $\Gamma$ of path constraints, and a path constraint $\varphi$, whether $\Gamma, a \models_{\text{(fin)}} \varphi$.

Let $\varphi = [R_{\mathcal{I}}]|(R_{\mathcal{I}} \subseteq R_\ell)$ be a path constraint. We call $\mathcal{I}$ a global model of $\varphi$, and write $\mathcal{I} \models \varphi$ if $\mathcal{I}^2 = \Delta^2$. We write $\mathcal{I} \models \Gamma$ for a set $\Gamma$ 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 $\Gamma \models_{\text{fin}} \varphi$ 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 $\Gamma$ of path constraints and a path constraint $\varphi$, whether $\Gamma \models_{\text{fin}} \varphi$.

**Example 9.** Consider the constraint $\varphi_1 = R_1 \subseteq R_2$, where
\[
R_1 = id(\text{Dept}) \circ \text{partOf}^{-} \circ \text{offers} - \text{requires} - \circ \text{partOf}^{-*} \\
R_2 = id(\text{Dept}) \circ \text{partOf}^{-*} \circ \text{offers}
\]
Intuitively, (a node interpreting) a department satisfies $\varphi_1$ 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 specific departments. For example, we may require it for computer science, and our example instance satisfies it: $I_{\text{Comp, CS Dept}} \models \varphi_1$. 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., $(id(\text{CS dept}) \cdot R_1) \subseteq R_2$.

To illustrate the use of prefixes, suppose that the policy expressed by $\varphi_1$ 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 that is required by a program within the School (while other schools may allow for mandatory courses that are offered by different departments). This is captured by the constraint with non-empty prefix $\varphi_1 = [id(\text{School SciEng}) \circ \text{hasDepartment}](R_1 \subseteq R_2)$.

Expressing Path Constraints in $ZOI$ with Role Difference. The path constraints we have introduced are not expressible in $ZOI$, neither under the pointed nor under the global semantics. (This is not surprising, since the former have an undecidable implication problem, as we will see below, while reasoning in $ZOI$ is decidable.) However, we can express path constraints very naturally if we extend $ZOI$ with difference $R \setminus R'$ of arbitrary roles, resulting in the logic we denote $ZOI\setminus$.

**Definition 10.** $ZOI\setminus$ roles are defined analogously to $ZOI$ roles, except that for complex roles we have $R, R' \rightarrow T \mid id(C) \mid S \mid R \cup R' \mid R \circ R' \mid R \setminus R' \mid R^*$. $ZOI\setminus$ concepts, assertions, axioms, and knowledge bases are defined as for $ZOI$, but allowing for $ZOI\setminus$ roles in the place of $ZOI$ roles. The semantics of $ZOI\setminus$ is also given as for $ZOI$.

The path constraints we have defined can be easily expressed in $ZOI\setminus$.

**Lemma 11.** For a path constraint $\varphi = [R_p]|(R_{\mathcal{I}} \subseteq R_\ell)$, let $C_\varphi = \forall R_p. (\forall (R_{\mathcal{I}} \setminus R_p). \bot)$. Then, for every instance $\mathcal{I}$, we have that $\varphi^\mathcal{I} = C_\varphi^\mathcal{I}$.

Thus (pointed) (finite) entailment of path constraints can be reduced to $ZOI\setminus$ KB unsatisfiability:

**Lemma 12.** For a set $\Gamma$ of path constraints, a path constraint $\varphi$, and $a \in N_\mathcal{I}$, we have that:
\[
\Gamma, a \models \varphi \quad \text{iff} \quad (\bigcap_{\gamma \in \Gamma} C_\gamma \cap \neg C_\varphi)(a) \quad \text{is unsatisfiable}, \\
\Gamma, a \models_{\text{fin}} \varphi \quad \text{iff} \quad (\bigcap_{\gamma \in \Gamma} C_\gamma \cap \neg C_\varphi)(a) \quad \text{is finitely unsatisfiable}, \\
\Gamma \models \varphi \quad \text{iff} \quad \bigwedge_{\gamma \in \Gamma} (T \subseteq C_\gamma) \land \neg (T \subseteq C_\varphi) \quad \text{is unsatisfiable}, \\
\Gamma \models_{\text{fin}} \varphi \quad \text{iff} \quad \bigwedge_{\gamma \in \Gamma} (T \subseteq C_\gamma) \land \neg (T \subseteq C_\varphi) \quad \text{is finitely unsatisfiable}.
\]
Verification of Evolving GSD under Expressive Path Constraints

We will see that, unfortunately, both implication of path constraints and reasoning in $ZOI$ 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 $ZOI$ extend to the implication of path constraints $\gamma$ where only simple $ZOI$ roles occur. Since in this case the resulting $C_\gamma$ is a $ZOI$ concept, from Lemma 12 and theorem 5 we get:

\textbf{Corollary 13.} Let $\Gamma$ be a set of path constraints and $\varphi$ a path constraint such that, for each $\gamma = \{R_p| (R_\ell \subseteq R_r) \in \Gamma \cup \{\varphi\}, R_p, R_\ell$ and $R_r$ are all simple $ZOI$ roles. Then $\Gamma, a \models \varphi$, $\Gamma, a \models_{\text{fin}} \varphi$, $\Gamma \models \varphi$, and $\Gamma \models_{\text{fin}} \varphi$ are all decidable in EXPSPACE.

\textbf{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:\footnote{In fact, the authors of [14] show that implication is r.e. complete, and finite implication co-r.e. complete.}

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

\begin{itemize}
  \item All prefixes, left tails, and right tails are one-way word roles (that is, only forward constraints according to [14] are allowed).
  \item All prefixes and left tails are one-way word roles different from $\varepsilon$, and each right tail is a one-way word role or an inverted one-way word role, and is different from $\varepsilon$.
\end{itemize}

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 Machine rather than a two-register machine as in [14], and we believe some readers may find it simpler.

\textbf{Theorem 15.} The problems of checking whether $\Gamma, a \models \varphi$ and whether $\Gamma, a \models_{\text{fin}} \varphi$, given $\Gamma$, $a$, and $\varphi$ are undecidable. This holds even when $\varphi$ is of the form $r_1 \subseteq r_2$ and $\Gamma$ contains only constraints of the following forms, where $\{r, r_1, r_2, r_3\} \subseteq N_R$:

\[ r_1 \circ r_2 \subseteq r_3 \quad r_1 \subseteq r_2 \circ r_3 \quad [r](r_1 \circ r_2 \subseteq r_3) \quad [r](r_1 \subseteq r_2 \circ r_3) \]

\textbf{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 $\Gamma, a \models \varphi_M$ as checking unsatisfiability of the first order formula that corresponds to $\Gamma$ and the negation of $\varphi_M$. The same is true for finite implication. Let $FO$ denote the set of FO formulae, $X$ be a recursive subset of $FO$, and let $f : FO \rightarrow X$ be a recursive function such that:

\begin{itemize}
  \item if $\beta \in FO$ is unsatisfiable, then $f(\beta)$ is unsatisfiable, and
  \item if $\beta \in FO$ has a finite model, then $f(\beta)$ has a finite model.
\end{itemize}

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 $\beta$ we can build a procedure that takes no input and terminates iff $\beta$ 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) $M$ with state set $Q$ and alphabet $\Sigma$. We assume that $M$ has two designated states $q_{\text{fin}}^o$ and $q_{\text{fin}}^r$. We build $\Gamma$ and $\varphi_M$ such that the following conditions are satisfied:
1. If $M$ reaches $q_{\text{fin}}$ starting from the empty string as input, then $\Gamma_M, a \models \varphi_M$.

2. If $M$ reaches $q_{\text{out}}$ starting from the empty string as input, then there exists a finite $I$ such that $I, a \models \Gamma_M$ and $I, a \not\models \varphi_M$.

We assume that $M$ 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 $\omega$. The transition function of $M$ is of the form $\delta : Q \times \Sigma \rightarrow \Sigma \times Q \times \{R, L\}$, with the usual reading, where R and L stand for right and left, respectively. The initial state of $M$ is $q_{\text{ini}}$.

The constraint $\varphi_M$ takes the form $u_{\text{ini}} \subseteq u_{\text{halt}}$ where $u_{\text{ini}}$ and $u_{\text{halt}}$ are two role names. We define the set $\Gamma_M$ next. Intuitively, the idea of the reduction is the following. Assume an arbitrary pointed structure $I, a$ and let $a$ denote $a^{\mathcal{I}}$. Whenever there is some $\alpha' \in A^{\mathcal{I}}$ such that $(a, \alpha') \in u_{\text{ini}}^{\mathcal{I}}$, the constraints in $\Gamma_M$ will ensure that if $I, a \models \Gamma_M$, then $I$ contains a (possibly infinite) structure that represents the computation of $M$, and that $(a, \alpha) \in u_{\text{halt}}^{\mathcal{I}}$ 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 $\Gamma_M$, which ensure that from the initial arc $u_{\text{ini}}$ we build a full computation of $M$. We use role names of the form $t_{q, \sigma}$ and $f_{q, \sigma}$ for each $q \in Q \cup \{\#\}$ and each $\sigma \in \Sigma \cup \{\omega\}$. Each of these symbols stands for a tape position containing the symbol $\sigma$. The marker $\#$ indicates that the head of $M$ is not on the current position, while $q \in Q$ indicates that the head is on the current position and $M$ 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 $u_{\text{aux}}$. They ensure that whenever there is an $u_{\text{ini}}$ arc, there exists also an arc labeled $f_{\text{aux}, \omega}$, indicating that $M$ is in state $q_{\text{ini}}$, the current tape position contains the symbol $\omega$, and the current tape position is the right-most one that has been visited so far:

$$u_{\text{ini}} \subseteq u_{\text{aux}} \circ u_{\text{out}}$$  
(1)

$$u_{\text{aux}} \subseteq u_{\text{ini}} \circ f_{\text{aux}, \omega}$$  
(2)

Note that there is an $u_{\text{ini}}$ arc from $o$ to the beginning of $f_{\text{aux}, \omega}$, and an $u_{\text{out}}$ arc from its end to $o'$.

For the initial and final points of this $f_{\text{aux}, \omega}$ arc, and of all the arcs $r$ that our construction will generate below in order to simulate the runs of $M$, we want the role name $u_{\text{ini}}$ to connect $o$ to the point, and $u_{\text{out}}$ to connect the point to $o'$. We ensure this by adding the following constraints:

$$[u_{\text{ini}} \circ r \subseteq u_{\text{ini}}] \quad \text{for every } r \in N_R \setminus \{u_{\text{ini}}, u_{\text{halt}}, u_{\text{ini}}, u_{\text{out}}, u_{\text{aux}}, \} \text{ occurring in } \Gamma_M$$  
(3)

$$[u_{\text{ini}} \circ u_{\text{out}} \subseteq u_{\text{out}}] \quad \text{for every } r \in N_R \setminus \{u_{\text{ini}}, u_{\text{halt}}, u_{\text{ini}}, u_{\text{out}}, u_{\text{aux}}, \} \text{ occurring in } \Gamma_M$$  
(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 $M$. With these axioms, the initial and final points of the $f_{\text{aux}, \omega}$ arc are both reachable 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 $M$ 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 next 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 ‘diagonal’ $d$ roles are used to propagate information between configurations.

The first group handles the case where the machine is currently 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:

$$[u_{\text{ini}}] \circ f_{q, \sigma} \subseteq d_{\#, \sigma'} \circ n_{f, q'}$$  
for each $\delta(q, \sigma) = (q', \sigma', R)$  
(5)

$$[u_{\text{ini}}] \circ n_{f, q} \subseteq f_{\omega, \omega} \circ d_f$$  
for each $q \in Q$  
(6)
### Verification of Evolving GSD under Expressive Path Constraints

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:

\[
[u_{in}](l_{q\sigma} \circ n_{\sigma} \subseteq d_{q\sigma}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{7}
\]

\[
[u_{in}](d_{q\sigma} \subseteq n_{\sigma} \circ t_{q\sigma}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{8}
\]

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

\[
[u_{in}](f_{q\sigma} \subseteq d_{qf} \circ n_{f}) \quad \text{for each } \delta(q, \sigma) = (q', \sigma', L) \tag{9}
\]

\[
[u_{in}](d_{qf} \circ n_{\sigma} \subseteq n_{q} \circ f_{\sigma}) \quad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\omega\} \tag{10}
\]

\[
[u_{in}](t_{\#\sigma} \circ n_{\sigma-q} \subseteq d_{q\sigma}) \quad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\omega\} \tag{11}
\]

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:

\[
[u_{in}](l_{q\sigma} \subseteq d_{-\#\sigma} \circ n_{q-f}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{12}
\]

\[
[u_{in}](l_{q-f} \circ n_{q} \subseteq d_{q\sigma}) \quad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\omega\} \tag{13}
\]

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):

\[
[u_{in}](l_{-q\sigma} \subseteq d_{q\sigma}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{14}
\]

\[
[u_{in}](d_{q\sigma} \subseteq t_{\sigma\#}n_{\sigma}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{15}
\]

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

\[
[u_{in}](n_{-\#} \circ f_{q\sigma} \subseteq d_{q\sigma f}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{16}
\]

\[
[u_{in}](d_{q\sigma f} \subseteq f_{q\sigma} \circ n_{f}) \quad \text{for each } q \in Q \cup \{\#, \sigma \in \Sigma \cup \{\omega\} \tag{17}
\]

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

\[
[u_{in}](l_{q\sigma} \subseteq d_{-q'\#\sigma'} \circ n_{-\sigma}) \quad \text{for each } \delta(q, \sigma) = (q', \sigma', L) \tag{18}
\]

\[
[u_{in}](d_{-q'\#\sigma'} \subseteq n_{-q} \circ t_{\#\sigma}) \quad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\omega\} \tag{19}
\]

\[
[u_{in}](t_{\#\sigma} \circ n_{-\sigma-q} \subseteq d_{q\sigma}) \quad \text{for each } q \in Q, \sigma \in \Sigma \cup \{\omega\} \tag{20}
\]

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

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
This concludes the proof of Theorem 15.

Theorem 16. The problems of deciding $\Gamma \models \varphi$ and $\Gamma \models_{\text{fin}} \varphi$ given $\Gamma$ and $\varphi$ are undecidable, even when $\varphi$ is of the form $r_1 \subseteq r_2$ and $\Gamma$ contains only constraints of the forms $r_1 \circ r_2 \subseteq r_3$ and $r_1 \subseteq r_2 \circ r_3$, for $\{r_1, r_2, r_3\} \subseteq \mathbb{N}_R$. 

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

Computation halts in $q_{\text{fin}}^1$. We first ensure that if an arc $t_{q_{\text{fin}}^1, \sigma}$ or $f_{q_{\text{fin}}^1, \sigma}$ for some $\sigma \in \Sigma \cup \{\bot\}$ was generated (that is, $q_{\text{fin}}^1$ was reached in the computation), an $u_{\text{halt}}$ arc is created; note that if $\mathcal{M}$ halts in $q_{\text{fin}}^1$ nothing enforces $u_{\text{halt}}$ and there will be a model of the constraints where $\varphi_{\mathcal{M}}$ does not hold.

Then we ensure that if there is an $u_{\text{halt}}$ 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 $u_{\text{in}}$ from $o$, and reach $o'$ via $u_{\text{in}}$:

\[
\begin{align*}
    u_{\text{in}} \circ t_{q_{\text{fin}}^1, \sigma} & \subseteq u_{\text{halt}} & \text{for each } \sigma \in \Sigma \cup \{\bot\} \\
    u_{\text{in}} \circ f_{q_{\text{fin}}^1, \sigma} & \subseteq u_{\text{halt}} & \text{for each } \sigma \in \Sigma \cup \{\bot\} \\
    u_{\text{halt}} \circ u_{\text{out}} & \subseteq u_{\text{halt}}
\end{align*}
\]  

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

1. $\mathcal{M}$ reaches $q_{\text{fin}}^1$ iff $\Gamma_{\mathcal{M}}, a \models \varphi_{\mathcal{M}}$.

Suppose $\mathcal{M}$ reaches $q_{\text{fin}}^1$. Assume an arbitrary $\mathcal{I}, a$ such that $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$, and assume there is an arbitrary $d \in \Delta^2$ such that $(a^2, d) \in u_{\text{fin}}^2$. Since $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ and $\mathcal{M}$ reaches $q_{\text{fin}}^1$, then $(a^2, d) \in u_{\text{halt}}^2$ follows. This implies $\mathcal{I}, a \models \varphi_{\mathcal{M}}$ and hence $\Gamma, a \models \varphi_{\mathcal{M}}$. For the converse, assume $\Gamma, a \models \varphi_{\mathcal{M}}$. 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 $d \in \Delta^2$ such that $(a^2, d) \in u_{\text{halt}}^2$ but $(a^2, d) \not\in u_{\text{halt}}^2$. Since $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$, by constraints (3), (4), (21), and (23) we have $(a^2, d) \not\in u_{\text{halt}}^2$, and $\mathcal{I}$ cannot contain any arcs of the form $t_{q_{\text{fin}}^1, \sigma}$ or $f_{q_{\text{fin}}^1, \sigma}$, which means that the computation of $\mathcal{M}$ does not reach the state $q_{\text{fin}}^1$.

2. If $\mathcal{M}$ reaches $u_{\text{halt}}^2$, 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 $q_{\text{fin}}^1$. Then this computation does not reach $q_{\text{fin}}^1$. Moreover, there is some $\mathcal{I}, a$ that simulates this computation, that is, such that $(a^2, d) \in u_{\text{fin}}^2$ and $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$. Since $\mathcal{M}$ does not reach $q_{\text{fin}}^1$, we can assume there are no arcs $t_{q_{\text{fin}}^1, \sigma}$ or $f_{q_{\text{fin}}^1, \sigma}$ in $\mathcal{I}$, and since the only constraints that imply $u_{\text{halt}}$ are (21) and (23), we can assume $u_{\text{halt}}^2 = \emptyset$.

Hence we have $\mathcal{I}, a \models \Gamma_{\mathcal{M}}$ with $(a^2, d) \in u_{\text{fin}}^2$ but $u_{\text{halt}}^2 = \emptyset$ as desired.

This concludes the proof of Theorem 15.

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 16. The problems of deciding $\Gamma \models \varphi$ and $\Gamma \models_{\text{fin}} \varphi$ given $\Gamma$ and $\varphi$ are undecidable, even when $\varphi$ is of the form $r_1 \subseteq r_2$ and $\Gamma$ contains only constraints of the forms $r_1 \circ r_2 \subseteq r_3$ and $r_1 \subseteq r_2 \circ r_3$, for $\{r_1, r_2, r_3\} \subseteq \mathbb{N}_R$. 

ICDT’16
Verification of Evolving GSD under Expressive Path Constraints

Proof. It suffices to observe that in the proof of Theorem 15, all constrains \([R_p]([R_\ell \subseteq R_r])\) with \(R_p \neq \emptyset\) have \(R_p = u_n\). The only role of this prefix is to make sure that \(R_\ell \subseteq R_r\) fires at all nodes of an instance, and not only at the interpretation of the given \(a\). 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 15 and 16 for path constraint implication, but one could just as well formulate them for the problem of deciding whether a fact \(r_1(a, b)\) and a set of constraints \(\Gamma\) that are satisfied (at \(a\) with the pointed semantics for the prefixed version, or everywhere for the prefix-empty version) entail that there is some 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 apply also to other constraint languages that can express \(\Gamma\). For instance, the prefix-empty version of \(\Gamma\) can be easily expressed in the restricted class of tuple generating dependencies of the forms:

\[
\begin{align*}
    r_1(x, y), r_2(y, z) &\to r_3(x, z) \\
    r_1(x, z) &\to \exists y. r_2(x, y), r_3(y, z)
\end{align*}
\]

Hence we have provided 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.

Turning back to DLs, the undecidability of path constraint implication implies the undecidability of (finite) satisfiability in \(\mathcal{ZOI}\). In fact, it implies the undecidability of (finite) satisfiability of a complex \(\mathcal{ZOI}\) role id(\(\forall R, \perp\)) of \((r_1 \setminus r_2)\), where \(R\) is a union of roles of the forms

\[
\begin{align*}
    (r_1 \circ r_2) \setminus r_3 &\quad r_1 \setminus (r_2 \circ r_3) &\quad r \circ ((r_1 \circ r_2) \setminus r_3) &\quad r \circ (r_1 \setminus (r_2 \circ r_3))
\end{align*}
\]

Undecidability also holds for the (finite) entailment of \(r_2(a, b)\) from \((id(\forall R, \perp) \circ r_1)(a, b)\). If we use a set \(K\) of concept inclusions of the form \(\top \cup \forall R_\ell \setminus R_r, \perp\) (where each \(R_\ell\) and \(R_r\) is the concatenation of at most two role names) to enforce prefix-empty constraints to hold everywhere, we obtain undecidability of deciding whether the KB \(K \land \{r_1(a, b)\}\) (finitely) entails \(r_2(a, b)\).

Finally, we remark that the constraints used in the proof above can also be written in GXPath\(_{reg}\), the extension of GXPath\(_{reg-pos}\) that allows for negation (and hence, intersection and difference) of path formulas. Hence Theorem 15 implies the undecidability of the (finite) implication of two GXPath\(_{reg}\) formulas \(\psi_1\) and \(\psi_2\), even when \(\psi_2\) is restricted to a role name \(r_2\) and \(\psi_1\) is a formula of the form \([-\xi] \circ r_1\) for \(r_1\) a role name and \(\xi\) 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 \(\varepsilon\).

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 decidable. As we show next, they 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 \(\mathcal{ZOI}\), showing decidability and an \(\text{EXPTIME}\) upper bound.

This is a significant improvement from the previous \(2\text{EXPSPACE}\) bound shown in [3] for the subclass of prefix-empty one-way regular path constraints.

\[\blacktriangleright\text{Theorem 17.}\] Let \(\Gamma\) be a set of prefix-empty path constraints, \(\varphi\) a prefix-empty path constraint, and \(a \in N_1\). Then we can obtain a set of \(\mathcal{ZOI}\) of axioms \(T_\varphi\) and a concept \(C_\varphi\) such that \(\Gamma \vdash \varphi\) iff \(\{T_\varphi\} \land \neg C_\varphi\) is unsatisfiable. Moreover, \(T_\varphi\) and \(C_\varphi\) can be constructed in linear time and the size of \(T_\varphi\) and \(C_\varphi\) are both linear, in the combined sizes of \(\Gamma\) and \(\varphi\).

Proof. For each \(\gamma = R_\ell \subseteq R_r \in \Gamma \cup \varphi\), let \(T_\gamma\) be the axiom \(\{a\} \subseteq \forall R_\ell . \exists \text{inv}(R_r).\{a\}\) where \(\text{inv}(R_r)\) is the inverted \(R_r\) defined as follows, where \(r \in N_\mathbb{R}\) and \(R, R'\) denote arbitrary roles:

\[
\begin{align*}
    \text{inv}(r) &= r^- \\
    \text{inv}(id(C)) &= id(C) \\
    \text{inv}(r \circ r') &= \text{inv}(R) \cap \text{inv}(R') \\
    \text{inv}(R \cap R') &= \text{inv}(R) \cup \text{inv}(R') \\
    \text{inv}(R \cup R') &= \text{inv}(R) \circ \text{inv}(R')
\end{align*}
\]
and \( \text{inv}(\varepsilon) = \varepsilon \). It is easy to see that \((d, d') \in R^\gamma \) iff \((d', d) \in \text{inv}(R)^\gamma \) for every \( d, d' \in \Delta^\gamma \).
It is then a straightforward consequence of the semantics of \( \text{ZOI} \) and of \( \gamma \), that for every pointed instance \( \mathcal{I}, a \) we have \( \mathcal{I}, a \models \gamma \) if and only if \( \mathcal{I} \models T \gamma \). Hence it easily follows that \( \Gamma \models \varphi \) if and only if \((\bigcup_{\gamma \in \Gamma} T \gamma) \land \neg T \varphi \) is unsatisfiable.

\[ \blacktriangleleft \]

From this reduction and Theorem 5 we get:

\[ \blacktriangledown \]

Corollary 18. The unrestricted and the finite pointed implication problems for prefix-empty path constraints are decidable in \( \text{EXPTIME} \).

We can combine this result and Corollary 13, obtaining that entailment is decidable in \( \text{EXPTIME} \) 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, \( \text{ZOI} \) can also express many other natural constraints, some of which are not easily expressible as path constraints. For example, \( \phi_5 \) in Example 3 does not directly correspond to a path constraint. \( \text{ZOI} \) 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:

\[
\text{Course} \sqcap \exists \text{requires}^{-} \sqsubseteq \text{UndergradProg} \sqcap \exists \text{teaches}^{-} \Join (\exists (\text{memberOf} \circ \text{partOf}^+) \sqsubseteq \text{Inst} \).
\]

To conclude this section, we remark that both the pointed and the global semantics are directly supported within \( \text{ZOI} \); 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 \( \text{ZOI} \) as a constraint language for GSD. We define below a language for manipulating GSD and show that it is possible to effectively reason about the preservation of \( \text{ZOI} \) constraints when data instances evolve as a result of operations in the proposed 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 \( \text{ZOI} \). 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 essentially the one in [4], but there the basic actions use concepts and roles in a different DL called \( \text{ALCHOIQ}_{br} \), instead of \( \text{ZOI} \). \( \text{ALCHOIQ}_{br} \) does not allow for regular expressions as roles, hence it can not express path queries and path constraints. On the other hand, it supports number restrictions, which are not allowed in \( \text{ZOI} \). We note that the language in [4] allows roles of the form \( R_{|C} \) (or \( \text{inv}(R)_{|C} \)), which stand for the pairs of objects in \( R \) whose first (resp., second) component is an instance of \( C \). Such roles are naturally expressible in \( \text{ZOI} \) using \( R \circ \text{id}(C) \).

Definition 19 (Action language). In what follows, additionally to \( N_I, N_C \) and \( N_R \), we consider a countably infinite set \( N_V \) of variables, disjoint from the other sets. We use \( \text{ZOI}_V \) concepts, roles, and KBs, which are defined as for \( \text{ZOI} \), 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 $t, t' \in N_{V} \cup N_{I}, A \in N_{C}, r \in N_{R} \setminus \{\emptyset\}$.

**Basics action** $\beta$ and **(complex) actions** $\alpha$ are defined by the following grammar:

$$
\beta \rightarrow (A \oplus C) \mid (A \ominus C) \mid (r \oplus R) \mid (r \ominus R) \\
\alpha \rightarrow \epsilon \mid \beta \cdot \alpha \mid (K ? \alpha[\alpha]) \cdot \alpha
$$

with $A$ a concept name, $C$ an arbitrary $ZOI_{V}$ concept, $r$ a role name, $R$ an arbitrary $ZOI_{V}$ role, and $K$ an arbitrary $ZOI_{V}$ knowledge base. The special symbol $\epsilon$ denotes the empty action.

We call a concept, role, knowledge base or an action *ground* if it has no variables. A substitution is a function $\sigma$ from $N_{V}$ to $N_{I}$. For a concept, role, knowledge base or an action $\gamma$, we use $\sigma(\gamma)$ to denote the result of replacing in $\gamma$ every occurrence of a variable $x$ by the individual $\sigma(x)$. An action $\alpha'$ is called a *ground instance* of an action $\alpha$ if $\alpha' = \sigma(\alpha)$ for some substitution $\sigma$.

Intuitively, an application of an action $(A \oplus C)$ on an instance $I$ stands for the addition of the content of $C^2$ to $A^2$. Similarly, $(A \ominus C)$ is used for the removal of $C^2$ from $A^2$. The two operations can also be performed on extensions of roles. Composition stands for successive action execution, and a conditional action $K ? \alpha_1[\alpha_2]$ expresses that $\alpha_1$ is executed if the instance is a model of $K$, and $\alpha_2$ 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 $K ? \alpha_1$, omitting $\alpha_2$.

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

**Definition 20** (Instance update, semantics of actions). Assume an instance $I$ and let $E$ be a concept or role name. If $E$ is a concept, let $W \subseteq \Delta^2$, otherwise, if $E$ is a role, let $W \subseteq \Delta^2 \times \Delta^2$. Then, $I \oplus E W$ (resp., $I \ominus E W$) denotes the instance $I'$ such that $\Delta^2 = \Delta^2 \cup W$ (resp., $\Delta^2 = \Delta^2 \setminus W$), and $E^2_1 = E^2_1$, for all symbols $E_1 \neq E$. Given a ground action $\alpha$, we define a mapping $S_{\alpha}$ from instances to instances as follows:

$$
S_{\alpha}(I) = I \\
S_{(A \oplus C)} \cdot \alpha(I) = S_{\alpha}(I \oplus_{A} C^2) \\
S_{(A \ominus C)} \cdot \alpha(I) = S_{\alpha}(I \ominus_{A} C^2) \\
S_{(r \oplus R)} \cdot \alpha(I) = S_{\alpha}(I \oplus_{r} R^2) \\
S_{(r \ominus R)} \cdot \alpha(I) = S_{\alpha}(I \ominus_{r} R^2)
$$

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 21.** Consider the following action with a free variable $x$, and its ground instance $\alpha_{dl}$:

$\alpha_{dl} = (\text{Course} \ominus \{x\}) \cdot (\text{requires} \ominus (\text{requires} \circ \text{id}((x)))) \cdot (\text{offers} \ominus (\text{offers} \circ \text{id}((x))))$

In $S_{\alpha_{dl}}(I_{BSc})$, we have the following changes, and the rest of the instance remains unchanged:

**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 $\alpha$ always preserves the satisfaction of integrity constraints given by a KB $K$. 

Definition 22 (The static verification problem). Let $\mathcal{K}$ be a KB. We say that an action $\alpha$ is $\mathcal{K}$-preserving if for every ground instance $\alpha'$ of $\alpha$ and every finite interpretation $\mathcal{I}$, we have that $\mathcal{I} \models \mathcal{K}$ implies $S_{\alpha'}(\mathcal{I}) \models \mathcal{K}$. The static verification problem consists on deciding, given an action $\alpha$ and a KB $\mathcal{K}$, whether $\alpha$ is $\mathcal{K}$-preserving.

Example 23. Recall the constraints $\mathcal{K}_{\text{uni}}$ from Ex. 3, and the action $\alpha_{dl}$ from Ex. 21. Note that $\alpha_{dl}$ is not $\mathcal{K}_{\text{uni}}$-preserving. In fact, this is witnessed by out instance $\mathcal{I}_{\text{uni}}$. We saw that in $S_{\alpha_{dl}}(\mathcal{I}_{\text{uni}})$ we have $(\text{MSc}_{\text{CompLogic}}, \text{mod}_{\text{KR}}) \in \text{requires}$, but $\text{mod}_{\text{KR}} \not\in (\exists \text{partOf}^{-*}. \text{Course})^2$, that is, the mandatory KR module does not contain any courses, violating $\phi_4 = T \sqsubseteq \exists \text{partOf}^{-*}. (\exists \text{partOf}^{-*}. \text{Course})$.

At the core of our technique for static verification is a transformation $\text{TR}_\alpha(\mathcal{K})$ that rewrites $\mathcal{K}$ incorporating the possible effects of an action $\alpha$. 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 $\alpha$ and $\mathcal{K}$. Intuitively, the models of $\text{TR}_\alpha(\mathcal{K})$ are exactly the interpretations $\mathcal{I}$ such that applying $\alpha$ 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 24. Given a $\mathcal{ZOI}$ KB $\mathcal{K}$, we use $\mathcal{K}_{E=E'}$ 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 $\alpha$, we define $\text{TR}_\alpha(\mathcal{K})$ as follows.

- $\text{TR}_\alpha(\mathcal{K}) = \mathcal{K}$
- $\text{TR}_{A\leftarrow B}(\mathcal{K}) = (\text{TR}_\alpha(\mathcal{K}))_{A\leftarrow A \leftarrow B}$
- $\text{TR}_{A\leftarrow B}(\mathcal{K}) = (\text{TR}_\alpha(\mathcal{K}))_{A\leftarrow A \leftarrow B \leftarrow C}$
- $\text{TR}_{A\leftarrow B}(\mathcal{K}) = (\text{TR}_\alpha(\mathcal{K}))_{A\leftarrow A \leftarrow B \leftarrow C}$
- $\text{TR}_{\text{nl}(\alpha_{dl})}(\mathcal{K}) = (\neg \mathcal{K} \lor \text{TR}_{\alpha_{dl}}(\mathcal{K})) \land (\mathcal{K} \lor \text{TR}_{\alpha_{dl}}(\mathcal{K}))$.

Assume a ground action $\alpha$ and a $\mathcal{ZOI}$ KB $\mathcal{K}$. Note that the transformation $\text{TR}_\alpha(\mathcal{K})$ may introduce role differences involving complex roles from $\alpha$. Hence $\text{TR}_\alpha(\mathcal{K})$ is a $\mathcal{ZOI}$ and need not be a $\mathcal{ZOI}$ KB. Note also that the size of $\text{TR}_\alpha(\mathcal{K})$ might be exponential in the size of $\alpha$. 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 $\mathcal{I} \models \text{TR}_\alpha(\mathcal{K})$. With the transformation $\text{TR}_\alpha(\mathcal{K})$ above we have a reduction from static verification to finite (un)satisfiability of $\mathcal{ZOI}$ KBs: an action $\alpha$ is not $\mathcal{K}$-preserving iff some finite model of $\mathcal{K}$ does not satisfy $\text{TR}_\alpha(\mathcal{K})$, where $\alpha^*$ is a ‘canonical’ grounding of $\alpha$. Formally, we have:

Theorem 25 ([4]). Assume a (complex) action $\alpha$ and a KB $\mathcal{K}$. Then the following are equivalent:

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

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 26. Deciding whether $\alpha$ is $\mathcal{K}$-preserving is undecidable, even when $\mathcal{K}$ is a trivial knowledge base of the form $\Gamma(a, b)$, and $\alpha$ is just a sequence of basic actions of the forms $(r \circ R)$ and $(r \oplus R)$, with $R$ a sequence of one or two concatenated role names.

Proof. We provide a reduction from deciding $\Gamma \models_{\text{fin}} r_1 \subseteq r_2$, where $\Gamma$ contains only constraints of the forms $r_1 \circ r_2 \subseteq r_3$ and $r_1 \subseteq r_2 \circ r_3$ for $r_1, r_2, r_3 \subseteq \text{ NR}$. We have seen above that this problem is undecidable. In particular, we construct an action $\alpha$ such that $\Gamma \models_{\text{fin}} r_1 \subseteq r_2$ iff $\alpha$ is $r_1(a, b)$-preserving. Let $R_1^1 \subseteq R_1^2, \ldots, R_n^1 \subseteq R_n^2$ be an enumeration of all constraints in $\Gamma$. For every $1 \leq i \leq n$, let $p_1^i$ and $p_2^i$ be fresh role names. Then $\alpha$ is the concatenation of the following actions in the given order:
Verification of Evolving GSD under Expressive Path Constraints

Theorem 27. \( \alpha \) in a ZOI see that the result of iteratively replacing role names by simple roles involving union and difference since the union and the difference of simple roles are simple roles, it is not hard to prove. Sketch. KB TR have that \( \alpha \) of size of \( \alpha \) to simulate \( r \) we get positive results if we restrict actions to simple roles.

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 \( r \mid id(C) \), and as our examples illustrate, this construct is quite useful. For nominals we can, however, simulate \( r \mid \{a\} \) in ZOI. We use a special role name \( T \mid \{a\} \) with the intended semantics \( (T \mid \{a\})^Z = \Delta^Z \times \{a\}^2 \), and write the simple role \( r \cap T \mid \{a\} \) in the place of \( r \mid \{a\} \). The intended meaning of \( T \mid \{a\} \) is easily enforced by adding the inclusion \( T \subseteq \exists T \mid \{a\} \) to any ZOI KB.

We call an action \( \alpha \) role-restricted if in every basic action of the form \( (r \cap R) \) we have that \( R \) is a simple \( ZOI \) role that may use the special role names \( T \mid \{a\} \).

Note \( \alpha_{id} \) in Example 21 can be rewritten as a role-restricted action as follows:
\[
\alpha_{id}' = (Course \cap \{DLSCS451\}) \cdot (\text{requires} \cap (\text{requires} \cap (T_{[DLSCS451]}))) \cdot (\text{offers} \cap (\text{offers} \cap (T_{[DLSCS451]}))) \cdot (\text{partOf} \cap (\text{partOf} \cap (T_{[DLSCS451]}))).
\]

\textbf{Theorem 27. Deciding whether \( \alpha \) is K-preserving for a given ZOI KB K and a role-restricted \( \alpha \text{ is EXP-TIME-complete.} \)

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 ZOI role results in a ZOI role. Hence, for any ZOI KB K and a role-restricted action \( \alpha \), the KB TR(\( \alpha \)) of KB but also a ZOI KB (i.e., difference is applied to simple roles only). Then from the decidability of (finite) satisfiability of ZOI it follows that checking whether \( \alpha \) is K-preserving is decidable. For the complexity upper bound, recall that the size of TR(\( \alpha \)) KB might be exponential in the size of \( \alpha \). However, as argued in [4], there are only exponentially many conjunctive clauses in disjunctive normal form from of \( K \land \neg TR(\alpha) \), each with size polynomial in the size of \( \alpha \) and \( K \). Thus from Theorems 5 and 25 we obtain the desired result.

5 Conclusions and Outlook

The main goal of this work was to advocate the use of the DL 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 ZOI where some roles must be interpreted as partial functions is a long standing open problem.

Acknowledgements This research has been partially supported by the EU IP project Optique (Scalable End-user Access to Big Data), grant agreement n. FP7-318338, by the Vienna Science and Technology Fund (WWTF), project ICT12-15, and by the Austrian Science Fund (FWF), projects P25207 and T515.
References

Verification of Evolving GSD under Expressive Path Constraints