Static Analysis of Graph Database Transformations

Iovka Boneva
iovka.boneva@univ-lille.fr
Univ. Lille, CNRS, UMR 9189 CRIStAL
F-59000 Lille, France

Benoît Groz
groz@lri.fr
Univ. Paris Saclay, CNRS, UMR 9015
LISN
91405 Orsay, France

Jan Hidders
j.hidders@bbk.ac.uk
Birkbeck, University of London
London, United Kingdom

Filip Murlak
f.murlak@uw.edu.pl
University of Warsaw
Warsaw, Poland

Sławek Staworko
slawek.staworko@relational.ai
RelationalAI
Berkeley, USA
Univ. Lille, CNRS, UMR 9189 CRIStAL
F-59000 Lille, France

ABSTRACT

We investigate graph transformations, defined using Datalog-like rules based on acyclic conjunctive two-way regular path queries (acyclic C2RPQs), and we study two fundamental static analysis problems: *type checking* and *equivalence* of transformations in the presence of graph schemas. Additionally, we investigate the problem of *target schema elicitation*, which aims to construct a schema that closely captures all outputs of a transformation over graphs conforming to the input schema. We show all these problems are in EXPTIME by reducing them to C2RPQ containment modulo schema; we also provide matching lower bounds. We use *cycle reversing* to reduce query containment to the problem of unrestricted (finite or infinite) satisfiability of C2RPQs modulo a theory expressed in a description logic.

CCS CONCEPTS

• **Theory of computation** → *Logic and databases.*

KEYWORDS

graph databases, static analysis, schemas, query containment

ACM Reference Format:
Iovka Boneva, Benoît Groz, Jan Hidders, Filip Murlak, and Sławek Staworko. 2023. Static Analysis of Graph Database Transformations. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3584372.3588654

1 INTRODUCTION

The growing adoption of graph databases calls for suitable data processing methods. Query languages for graph databases typically define their semantics as a set of tuples, which alone is inadequate for scenarios such as (materialized) graph database views and data migration in the context of schema evolution [11], with the schema describing the expected structure of the graph. A more adequate mechanism is that of a *transformation*, which takes a graph as input and produces a graph on the output.

Example 1.1. Consider a scenario where the schema of a medical knowledge graph undergoes changes due to advances in the understanding of biomolecular processes. The purpose of this knowledge graph is to catalog vaccines based on the antigen they are designed to target and to identify the pathogens that exhibit the antigens, each antigen being exhibited by at least one pathogen. Additionally, some pairs of antigens are known to be *cross reacting*: if a vaccine v targets an antigen x that is cross reacting with an antigen y, then v also targets y. Thus, the set of all antigens targeted by a vaccine is represented implicitly.

The schema S_0 of the original knowledge graph is presented in Figure 1 as a graph itself. It specifies the allowed node and edge

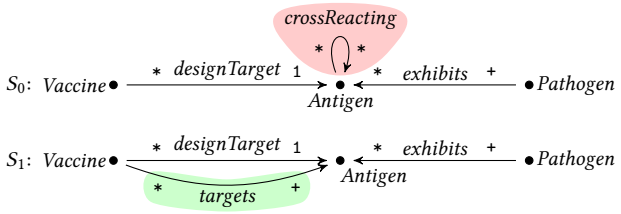

Figure 1: Evolving schema of a medical knowledge graph.

labels, and expresses participation constraints on edges in a manner that is typical for data modeling languages, e.g., $A \xrightarrow{* \; r \; 1} B$ indicates that every A-node has one outgoing r-edge to a B-node but a B-node may have arbitrarily many incoming r-edges from A-nodes.

Now, suppose that new findings refute the rule of cross-reactivity of antigens. The *cross-reacting* edges between antigens are no longer adequate for representing information about the antigens that a vaccine targets, and so, in the new schema S_1, this information is recorded explicitly with *targets* edges. Since up to that point, the knowledge graph did not contain any data points that contradicted the cross-reactivity rule, the logic of the rule can be used to transform the old knowledge graph to one that conforms to the new schema. Afterwards *cross-reacting* edges are removed. □

In the present paper, we study two classical problems of static analysis on graph transformations: *type checking*, that verifies if for every graph conforming to the source schema the transformation

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06.
https://doi.org/10.1145/3584372.3588654

outputs a graph conforming to the target schema, and *equivalence*, that verifies if two transformations produce the same output for every graph conforming to the source schema. Additionally, when the target schema is not known, we investigate the problem of *target schema elicitation* that constructs the containment-minimal target schema that captures the graphs produced by the transformation.

We study *executable* graph transformations defined with Datalog-like rules. The rules specify how to construct the output graph from the results of regular path queries evaluated over the input graph. To allow multiple copies of the same input node the rules use *node constructors*, essentially explicit Skolem functions that create nodes. As an example, the cross-reactivity rule from Example 1.1 gives rise to the following graph transformation rule

$$targets(f_V(x), f_A(y)) \leftarrow (designTarget \cdot crossReacting^*)(x, y)\,,$$

where $f_V(x)$ and $f_A(y)$ are constructors of *Vaccine* and *Antigen* nodes respectively. The two constructors can, for instance, have the following definitions $f_V(x) = (Vaccine, x)$ and $f_A(y) = (Antigen, y)$; essentially, they take the identifiers of the original nodes and decorate them with their type.

We investigate transformations that use only *acyclic two-way conjunctive regular path queries* (acyclic C2RPQs), which is arguably of practical relevance in the context of graph transformations. For instance, we have found no cyclic queries in the transformations implementing graph data migration between consecutive versions of the FHIR data format [34, 53] (Fast Healthcare Interoperability Resources is an international standard for interchange of medical healthcare data). Our constructions rely on acyclicity of C2RPQs to obtain relatively low computational complexity. We argue that the acyclicity assumption cannot be lifted without a significant complexity increase (see Section 7).

Node constructors are closely related to object creating functions [36, 37]. Our use of node constructors is inspired by analogous constructions in transformation languages such as R2RML [18, 22, 55], where node IRIs are typically obtained by concatenation of a URL prefix and the key values of a object represented by the constructed node. Our node constructors can have an arbitrary arity, thus allowing for instance to create nodes in the target graph that represent relationships (edges) between nodes in the source graph. To isolate the concern of possible overlaps between node constructors, we make the natural assumption that node constructors are injective, have pair-wise disjoint ranges, and for every node kind (label) a single dedicated node constructor is used. These assumptions allow us to remove the need to analyze the definitions of node constructors, which is out of the scope of the present paper, and they are consistent with how the analogous constructions are used in languages such as R2RML and FHIR mapping language.

For schemas, we employ a natural formalism of *graph schemas with participation constraints*, inspired by standard data modeling languages such as Entity-Relationship diagrams [17], and already studied, for instance, in the context of graph database evolution [11]. Such schemas allow one to declare the available labels of nodes and edges and to express participation constraints. In contrast to more expressive languages as ShEx and SHACL [19, 57], our formalism allows a *single label per node*, which determines the node type. Thus, roughly speaking, our schema formalism is to ShEx and SHACL what DTD is to XML Schema.

The key contributions of the present paper are as follows.

(1) We define graph database transformations and we reduce the problems of interest to *containment* of C2RPQs in unions of acyclic C2RPQs *modulo schemas*.
(2) We reduce the query containment problem to the unrestricted (finite or infinite) satisfiability of a C2RPQ modulo a set of constraints expressed in the Horn fragment of a description logic known as $\mathcal{ALCIF}$. The reduction involves an application of the *cycle reversing* technique [20, 38], carefully tailored to our needs.
(3) The unrestricted satisfiability problem for $\mathcal{ALCIF}$ can be solved in EXPTIME owing to a simple model property [16], but applying this result directly to the instance obtained via cycle reversing would lead to doubly exponential complexity due to an exponential blow-up inherent to cycle reversing. We provide a new algorithm with *improved complexity bounds*, which allows to accommodate the blow-up while keeping the overall complexity in EXPTIME. We also reformulate the simplicity of models in terms of a graph-theoretical notion of (k, l)-*sparsity* [44], which allows to streamline the reasoning.

These reductions allow to solve all problems of interest in EXPTIME and we also establish the matching lower bounds.

The paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce basic notions. In Section 4 we define graph transformations and the problems of interest, which we reduce to query containment modulo schema. In Section 5 we reduce the latter to satisfiability of a query modulo Horn-$\mathcal{ALCIF}$ theory, which we solve in Section 6. In Section 7 we summarize our findings and identify directions of future work. A technical report containing full proofs can be found in [9].

2 RELATED WORK

Graph-based data models have been proposed in various forms and shapes since the 1980s [4]. The proposals in the 1980s and 1990s included labeled graphs [32], graphs where certain nodes represent complex values [33, 43], graphs where nodes have associated complex values [1, 2], and graphs where nodes are associated with nested graphs [45]. More recently the RDF data model [30] and the Property Graph data model [3] have become popular. RDF graphs are similar to labeled graphs except that nodes are unlabeled and participate in at least one edge, and the labels of edges can be nodes and participate in edges. Property Graphs are also similar to labeled graphs except that nodes and edges have multiple labels and properties, and edges have identity. In our work we assume one of the simplest models, namely, labeled graphs where nodes have multiple labels and edges have a single label; our schemas require exactly one label per node. Since we focus here on transformations of the graph structure, we have no explicit notion of value associated with nodes and edges, but there are straightforward ways of adding this, as is done for example in [32].

The term **graph transformations** can refer to different formalisms [54]: the purpose of graph grammars is to define graph languages; algebraic graph transformations are mainly used to model systems with infinite behavior and are not functional (they produce multiple outputs on single input). Therefore, not only are

these formalisms ill-suited for defining transformations of graph databases, but also the problems studied for them are unrelated to the problems we study here. Monadic second-order (MSO) graph transductions [21] can capture our transformations only when restricted to unary node constructors; moreover, resorting to MSO logic typically incurs a prohibitive complexity overhead.

Transformation languages for graph databases are often based on Datalog extended with node-creation syntax in the head of the rules. It could be just a variable that is not bound in the body of the rule, like in IQL [2] and G-Log [51]; this ensures a fresh node is created for each valuation that makes the body true. Another option is to replace the unbound variable with a term consisting of a constructor function (sometimes called a Skolem function) applied to bound variables, like in O-logic [46] and F-logic [41]; the constructor creates a fresh node when called for the first time for certain arguments, and after that the same node for the same arguments. We adopt the idea of node constructors because we believe it provides a powerful and intuitive way to control the identity of new nodes. A different proposal, based on structural recursion, is offered by UnQL [12], but the underlying data model considers graphs equivalent if they are bisimilar, which makes the expressive power quite different. Graph transformations can also be expressed using query languages such as SPARQL and Cypher. Nevertheless, we believe that a rule-based transformation language is more convenient for defining transformations and it can co-exist with an expressive query language. For instance, in the XML world, XSLT [40] (rule-based) focuses on transformations, while XQuery [56] is mostly used for querying XML data. In the context of data exchange, **schema mappings** provide a declarative way to define database transformations [7, 13, 24]. Our transformations could be simulated by considering canonical solutions for plain SO-tgds [5] extended to allow acyclic C2RPQs in rule bodies. Note, however, that equivalence is undecidable for plain SO-tgds with keys [25], and open for plain SO-tgds [42].

The **static type checking problem** originates in formal language theory and has been studied for finite state transducers on words and for various kinds of tree transducers, including some designed to capture XML transformation languages [47–50]. Type checking has also been studied for graph transformations. In [33] labelled graphs are transformed using addition, deletion, and reduction operations, and type checking is investigated for schemas similar to ours but without participation constraints. The typing problem for UnQL is studied in [39], but the approach relies on schemas specifying graphs up to bisimulation, which limits their power to express participation constraints. Regarding transformations defined by schema mappings, if the mapping does not define target constraints, then the target schema is simply a relational signature and type checking is reduced to trivial syntactic check, and as such it is irrelevant. This is most often the case for graph schema mappings [7, 13], with seldom exceptions such as [10] for mapping relational to graph-shaped data. Their notion of consistency is related to type checking, but is studied for a simpler formalism without path queries. In the context of XML schema mappings, absolute consistency can be seen as a counterpart of type checking for non-functional transformations [8].

3 PRELIMINARIES

Graphs. We fix an enumerable set $\mathcal{N}$ of node identifiers, a recursively enumerable set Γ of node labels, and an recursively enumerable set Σ of edge labels. We work with labeled directed graphs, and in general, a node may have multiple labels while an edge has precisely one label. We allow, however, multiple edges between the same pair of nodes, as long as these edges have different labels. We model graphs as relational structures over unary relation symbols Γ and binary relation symbols Σ. That is, a graph G is a pair $\left(dom(G), \cdot^G\right)$ where $dom(G) \subseteq \mathcal{N}$ is the set of nodes of G and the function $\cdot^G$ maps each $A \in \Gamma$ to a set $A^G \subseteq dom(G)$ and each $r \in \Sigma$ to a binary relation $r^G \subseteq dom(G) \times dom(G)$. A graph G is *finite* if $dom(G)$ is finite and A^G and r^G are empty for all but finitely many $A \in \Gamma$ and $r \in \Sigma$. In the sequel, we use $u, v, \ldots$ to range over node identifiers, $A, B, C, \ldots$ to range over node labels, and $r, r', \ldots$ to range over edge labels. Also, we use r^- for inverse edges and let $(r^-)^G = \left\{(u,v) \mid (v,u) \in r^G\right\}$. We let $\Sigma^\pm = \Sigma \cup \{r^- \mid r \in \Sigma\}$ and use $R, R', \ldots$ to range over $\Sigma^\pm$.

Schemas. We consider a class of schemas that constrain the number of edges between nodes of given labels and we express these constraints with the usual symbols: `?` for at most one, `1` for precisely one, `+` for at least one, `*` for arbitrary many, and `0` for none. We focus on these basic cardinality constraints that are most commonly used in practice; e.g., Chen's original ER diagrams only used those [17]. In fact, we were unable to find any non-basic cardinality constraints in the FHIR specifications [34], while in the SHACL schemas in Yago 4.0 [58] we found only one: a person may have at most two parents.

Now, a *schema* is a triple $S = (\Gamma_S, \Sigma_S, \delta_S)$, where $\Gamma_S \subseteq \Gamma$ is a finite set of allowed node labels, $\Sigma_S \subseteq \Sigma$ is a finite set of allowed edge labels, and $\delta_S : \Gamma_S \times \Sigma_S^\pm \times \Gamma_S \to \{\mathtt{?}, \mathtt{1}, \mathtt{+}, \mathtt{*}, \mathtt{0}\}$. Schemas can be presented as graphs themselves, interpreted as illustrated next.

Example 3.1. Take the schema S_0 in Figure 1 and consider, for instance, the *designTarget* edge. It indicates that every *Vaccine* has a single design target *Antigen*, in symbols

$$\delta_{S_0}(\mathit{Vaccine}, \mathit{designTarget}, \mathit{Antigen}) = \mathtt{1}\,,$$

and that every *Antigen* may be the design target of an arbitrary number of *Vaccine*s, in symbols

$$\delta_{S_0}(\mathit{Antigen}, \mathit{designTarget}^-, \mathit{Vaccine}) = \mathtt{*}\,.$$

Edges that are not present are implicitly forbidden, e.g., no *exhibits* edge is allowed from *Vaccine* to *Pathogen*:

$$\delta_{S_0}(\mathit{Vaccine}, \mathit{exhibits}, \mathit{Pathogen}) = \mathtt{0}\,,$$
$$\delta_{S_0}(\mathit{Pathogen}, \mathit{exhibits}^-, \mathit{Vaccine}) = \mathtt{0}\,. \qquad \square$$

Now, a graph G *conforms* to a schema S if 1) every node in G has a single node label in Γ_S and every edge has a label in Σ_S, and 2) for all $A, B \in \Gamma_S$ and $R \in \Sigma_S^\pm$, for every node with label A the number of its R-successors with label B is as specified by $\delta_S(A, R, B)$. By $L(S)$ we denote the set of all *finite* graphs that conform to S.

Queries. We work with conjunctive two-way regular path queries (C2RPQs) that have the form

$$q(\bar{x}) = \exists \bar{y}.\varphi_1(z_1, z_1') \wedge \ldots \wedge \varphi_k(z_k, z_k')\,,$$

where $\bar{x} = \{z_1, z'_1, \ldots, z_k, z'_k\} \setminus \bar{y}$ and for every $i \in \{1, \ldots, k\} z_i$ and z'_i are variables and the formula φ_i is a regular expression that follows the grammar

$$\varphi ::= \varnothing \mid \epsilon \mid A \mid R \mid \varphi \cdot \varphi \mid \varphi + \varphi \mid \varphi^*,$$

where $A \in \Gamma$ matches nodes, $R \in \Sigma^{\pm}$ matches edges, ϵ matches empty paths, and $\varnothing$ matches no path. The semantics of C2RPQs is defined in the standard fashion [15] and we denote the set of *answers* to $q(\bar{x})$ in G by $[q(\bar{x})]^G$.

Example 3.2. Recall the schema S_0 in Figure 1. The following query selects vaccines together with the antigens they are designed to target or target through cross-reaction.

$$q(x, y) = (\mathit{Vaccine} \cdot \mathit{designTarget} \cdot \mathit{crossReacting}^* \cdot \mathit{Antigen})(x, y). \quad \square$$

Trivial atoms are of the form $\varnothing(x, x)$, $\epsilon(x, x)$, and $A(x, x)$, and in the sequel, we abuse notation and write them as unary atoms: $\varnothing(x)$, $\epsilon(x)$, and $A(x)$, respectively. The *multigraph* of a C2RPQ q has variables of q as nodes and an edge from x to y for every non-trivial atom $\varphi(x, y)$. The subclass of *acyclic* C2RPQs consists of queries whose multigraph is acyclic i.e., it does not have a path consisting of distinct edges that visits the same node twice. Note that acyclicity for C2RPQs needs to be more restrictive than the classical acyclicity of conjunctive queries based on Gaifman graphs. Indeed, the Gaifman graph of a C2RPQ $\varphi(x, y) \wedge \psi(x, y)$ is acyclic but its matches may form nontrivial cycles in the input graph.

A *Boolean* C2RPQ q has all its variables existentially quantified, and it may have only a single answer, the empty tuple, in which case, we say that q is *satisfied* in G and write $G \models q$. We also use *unions* of C2RPQs (abbreviated as UC2RPQs) represented as sets of C2RPQs $Q(\bar{x}) = \{q_1(\bar{x}), \ldots, q_k(\bar{x})\}$ and extend the notions of answers, satisfaction, and acyclicity to UC2RPQs in the natural fashion. Given two UC2RPQs $P(\bar{x})$ and $Q(\bar{x})$, and a schema S, we say that $P(\bar{x})$ *is contained in* $Q(\bar{x})$ *modulo* S, in symbols $P(\bar{x}) \subseteq_S Q(\bar{x})$, if $[P(\bar{x})]^G \subseteq [Q(\bar{x})]^G$ for every $G \in L(S)$.

Description logics. We operate on properties of graphs formulated in the *description logic* $\mathcal{ALCIF}$ (and its fragments) [6]. In description logics, elements of Γ and Σ are called *concept names* and *role names*, respectively. $\mathcal{ALCIF}$ allows to build more complex concepts with the following grammar:

$$C ::= \bot \mid A \mid C \sqcap C \mid \neg C \mid \exists R.C \mid \exists^{\leq 1} R.C,$$

where $A \in \Gamma$ and $R \in \Sigma^{\pm}$. We also use additional operators that are redundant but useful when defining fragments; for brevity we introduce them as syntactic sugar: $\top := \neg\bot$, $C_1 \sqcup C_2 := \neg(\neg C_1 \sqcap \neg C_2)$, $\forall R.C := \neg\exists R.\neg C$, $\nexists R.C := \neg\exists R.C$. We extend the interpretation function $\cdot^G$ to complex concepts as follows:

$$\bot^G = \emptyset, \quad (C_1 \sqcap C_2)^G = C_1^G \cap C_2^G, \quad (\neg C)^G = \mathit{dom}(G) \setminus C^G,$$
$$(\exists R.C)^G = \left\{u \in \mathit{dom}(G) \mid \exists v.\ (u, v) \in R^G \wedge v \in C^G\right\},$$
$$(\exists^{\leq 1} R.C)^G = \left\{u \in \mathit{dom}(G) \mid \exists^{\leq 1} v.\ (u, v) \in R^G \wedge v \in C^G\right\}.$$

Statements in description logics have the form of *concept inclusions*,

$$C \sqsubseteq D$$

where C and D are concepts. A graph G *satisfies* $C \sqsubseteq D$, in symbols $G \models C \sqsubseteq D$, if $C^G \subseteq D^G$. A set $\mathcal{T}$ of concept inclusions is traditionally called a *TBox* and we extend satisfaction to TBoxes in the canonical fashion: $G \models \mathcal{T}$ if $G \models C \sqsubseteq D$ for each $C \sqsubseteq D \in \mathcal{T}$.

In the *Horn fragment* of $\mathcal{ALCIF}$, written Horn-$\mathcal{ALCIF}$, we only allow concept inclusions in the following normal forms:

$$K \sqsubseteq A, \qquad K \sqsubseteq \bot, \qquad K \sqsubseteq \forall R.K',$$
$$K \sqsubseteq \exists R.K', \qquad K \sqsubseteq \nexists R.K', \qquad K \sqsubseteq \exists^{\leq 1} R.K',$$

where $A \in \Gamma$, $R \in \Sigma^{\pm}$, and K, K' are intersections of concept names (intersection of the empty set of concepts is $\top$). If statements of the form $K \sqsubseteq A_1 \sqcup A_2 \sqcup \cdots \sqcup A_n$ are allowed too, then we recover the full power of $\mathcal{ALCIF}$ (up to introducing auxiliary concept names).

Participation constraints of schemas can be expressed with simple Horn-$\mathcal{ALCIF}$ statements as illustrated in following example.

Example 3.3. For instance, the assertion in S_0 (Figure 1) that *Pathogen* manifests at least one *Antigen* is expressed with the statement $\mathit{Pathogen} \sqsubseteq \exists \mathit{exhibits}.\mathit{Antigen}$. The assertion that an *Antigen* may be exhibited by an arbitrary number of *Pathogens* needs no Horn-$\mathcal{ALCIF}$ statement. However, statements are needed for implicitly forbidden edges, e.g., $\mathit{Vaccine} \sqsubseteq \nexists \mathit{exhibits}.\mathit{Antigen}$. $\square$

4 GRAPH TRANSFORMATIONS

We propose transformations of graphs defined with Datalog-like rules that use acyclic C2RPQs in their bodies. To allow multiple copies of the same source node we use node constructors. Formally, a *k-ary node constructor* is a function $f : \mathcal{N}^k \to \mathcal{N}$ and we denote the set of node constructors by $\mathcal{F}$. To remove the concern of overlapping node constructors, and the need to analyze their definitions, we assume that for every node label $A \in \Gamma$ we have precisely one node constructor f_A, all node constructors are injective, and their ranges are pairwise disjoint.

We introduce two kinds of *graph transformation rules*: node rules and edge rules. A *node rule* has the form

$$A\big(f_A(\bar{x})\big) \leftarrow q(\bar{x}),$$

where $A \in \Gamma$, $f_A \in \mathcal{F}$, and q is an acyclic C2RPQ. An *edge rule* has the form

$$r\big(f(\bar{x}), f'(\bar{y})\big) \leftarrow q(\bar{x}, \bar{y}),$$

where $r \in \Sigma$, $f, f' \in \mathcal{F}$, and q is an acyclic C2RPQ. Note that an equality between variables $z = z'$ can be expressed as $\epsilon(z, z')$, and consequently, we can assume that $\bar{x}$ and $\bar{y}$ are disjoint.

Now, a *graph transformation* T is a finite set of graph transformation rules. By Γ_T and Σ_T we denote the finite sets of node and edge labels, respectively, used in the heads of the rules of T.

Example 4.1. Below we present rules defining the transformation T_0 of the medical database, described in Example 1.1. We use 3 unary node constructors $f_A(x)$ for *Antigen* nodes, $f_P(x)$ for *Pathogen* nodes, and $f_V(x)$ for *Vaccine* nodes.

$$\begin{aligned}
\mathit{Vaccine}(f_V(x)) &\leftarrow (\mathit{Vaccine})(x),\\
\mathit{Antigen}(f_A(x)) &\leftarrow (\mathit{Antigen})(x),\\
\mathit{designTarget}(f_V(x), f_A(y)) &\leftarrow (\mathit{designTarget})(x, y),\\
\mathit{targets}(f_V(x), f_A(y)) &\leftarrow (\mathit{designTarget} \cdot \mathit{crossReacting}^*)(x, y),\\
\mathit{Pathogen}(f_P(x)) &\leftarrow (\mathit{Pathogen})(x),\\
\mathit{exhibits}(f_P(x), f_A(y)) &\leftarrow (\mathit{exhibits})(x, y).
\end{aligned}$$
$\square$

Now, given a graph G and a graph transformation T the *result of applying T to G* is a graph $T(G)$ such that (for $A \in \Gamma$ and $r \in \Sigma$)

$$A^{T(G)} = \{f_A(t) \mid A(f_A(\bar{x})) \leftarrow q(\bar{x}) \in T,\ t \in [q(\bar{x})]^G\},$$
$$r^{T(G)} = \{(f(t), f'(t')) \mid r(f(\bar{x}), f'(\bar{y})) \leftarrow q(\bar{x}, \bar{y}) \in T, (t, t') \in [q(\bar{x}, \bar{y})]^G\}.$$

We are interested in the following two classical static analysis tasks.

Type checking Given a transformation T, a source schema S, and a target schema S' check whether for every G that conforms to S the output of transformation $T(G)$ conforms to S'.

Equivalence Given a source schema S and two transformations T_1 and T_2 check whether T_1 and T_2 agree on every graph that conforms to S.

In settings where the target schema is not known, it might be useful to construct one. Naturally, we wish to preclude a trivial solution that produces the universal schema that accepts all graphs over a given set of node and edge labels. Instead, we propose to construct a schema that offers the tightest fit to the set of output graphs. To define formally this requirement, we define schema containment in the classical fashion: a schema S is contained in S' if and only if $L(S) \subseteq L(S')$.

Schema elicitation Given a transformation T and a source schema S, construct the containment-minimal target schema S' such that $T(G) \in L(S')$ for every $G \in L(S)$.

We observe that $T(G)$ may have nodes with no label, which may preclude it from satisfying any schema, and consequently, schema elicitation may also return error.

We prove the main result by reducing the problems of interest to query containment modulo schema (and vice versa), which we later show to be EXPTIME-complete. Although schema elicitation is not a decision problem, we show EXPTIME-completeness of deciding if the result of schema elicitation is equivalent to a given schema. Should schema elicitation have lesser complexity, so would have the corresponding decision problem since schema equivalence is easily decided in polynomial time.

Theorem 4.2. *Type checking, schema elicitation, and equivalence of graph transformations are* EXPTIME*-complete.*

We outline the main ideas of the proof by illustrating how a transformation T can be analyzed with a toolbox of methods based on query containment modulo source schema S. We formulate these methods with an entailment relation:

$$(T, S) \models K \sqsubseteq K' \quad \textit{iff} \quad T(G) \models K \sqsubseteq K' \text{ for every } G \in L(S).$$

W.l.o.g. we assume that every rule of transformation T is *trim* i.e., it uses in its body a query $q(\bar{x})$ that is satisfiable modulo S, in symbols $\exists \bar{x}.q(\bar{x}) \not\subseteq_S \varnothing$; otherwise, T can be trimmed.

First, we group queries from rules of T based on the labels of nodes and edges they create. For $A, B \in \Gamma_T$ and $r \in \Sigma_T$ we define

$$Q_A(\bar{x}) = \{q(\bar{x}) \mid A(f_A(\bar{x})) \leftarrow q(\bar{x}) \in T\},$$
$$Q_{A,r,B}(\bar{x}, \bar{y}) = \{q(\bar{x}, \bar{y}) \mid r(f_A(\bar{x}), f_B(\bar{y})) \leftarrow q(\bar{x}, \bar{y}) \in T\},$$
$$Q_{A,r^-,B}(\bar{x}, \bar{y}) = \{q(\bar{y}, \bar{x}) \mid r(f_B(\bar{y}), f_A(\bar{x})) \leftarrow q(\bar{y}, \bar{x}) \in T\}.$$

In essence, $Q_A(\bar{x})$ identifies tuples over the input graph that yield a node constructed with f_A and with label A while $Q_{A,R,B}(\bar{x}, \bar{y})$ identifies tuples that yield R-edges from a node created with f_A to a node created with f_B.

Example 4.3. A couple of examples of above queries for the transformation T_0 in Example 4.1 follow.

$$Q_{Vaccine}(x) = (Vaccine)(x),$$
$$Q_{Vaccine,targets,Antigen}(x, y) = (designTarget \cdot crossReacting^*)(x, y),$$
$$Q_{Vaccine,designTarget,Antigen}(x, y) = (designTarget)(x, y).$$ □

Since an edge rule does not assign labels to nodes it creates, the result of a transformation may be a graph with nodes without a label. Such a situation precludes type checking from passing and prevents schema elicitation from producing meaningful output. Consequently, we first verify that every node in every output graph has exactly one label, in symbols $(T, S) \models \top \sqsubseteq \bigsqcup \Gamma_T$, where $\bigsqcup\{A_1, \ldots, A_k\}$ is a shorthand for $A_1 \sqcup \ldots \sqcup A_k$. We prove the following.

$$(T, S) \models \top \sqsubseteq \bigsqcup \Gamma_T \quad \textit{iff}$$
$$\exists \bar{y}.Q_{A,R,B}(\bar{x}, \bar{y}) \subseteq_S Q_A(\bar{x}) \quad \text{for all } A, B \in \Gamma_T \text{ and } R \in \Sigma_T^{\pm}.$$

We point out that the restriction of one node constructor per node label ensures that each node of the output has at most one label.

Example 4.4. Take T_0 from Example 4.1 and S_0 in Figure 1. Verifying that $(T_0, S_0) \models \top \sqsubseteq \bigsqcup \Gamma_{T_0}$ requires a number of containment tests including the following two.

$$\exists y.(designTarget \cdot crossReacting^*)(x, y) \subseteq_{S_0} (Vaccine)(x),$$
$$\exists y.(designTarget)(x, y) \subseteq_{S_0} (Vaccine)(x).$$ □

Now, to perform type checking against a given target schema S', we verify that $\Gamma_T \subseteq \Gamma_{S'}$ and $\Sigma_T \subseteq \Sigma_{S'}$. Then, we take the TBox $\mathcal{T}_{S'}$ of concept inclusions that expresses participation constraints of the target schema S' and we verify that $(T, S) \models \mathcal{T}_{S'}$. Type checking succeeds if and only if all the above tests succeed.

The TBox $\mathcal{T}_{S'}$ consists of statements from a small fragment $\mathcal{L}_0$ of Horn-$\mathcal{ALCIF}$ which allows only statements of the forms

$$A \sqsubseteq \exists R.B, \qquad A \sqsubseteq \nexists R.B, \qquad A \sqsubseteq \exists^{\leq 1} R.B,$$

where $A, B \in \Gamma$ and $R \in \Sigma^{\pm}$. The entailment of such statements is also reduced to query containment:

$$(T, S) \models A \sqsubseteq \exists R.B \quad \textit{iff} \quad Q_A(\bar{x}) \subseteq_S \exists \bar{y}.Q_{A,R,B}(\bar{x}, \bar{y}),$$
$$(T, S) \models A \sqsubseteq \nexists R.B \quad \textit{iff} \quad \exists \bar{y}.Q_A(\bar{x}) \wedge Q_{A,R,B}(\bar{x}, \bar{y}) \subseteq_S \bigwedge_i \varnothing(x_i),$$
$$(T, S) \models A \sqsubseteq \exists^{\leq 1} R.B \quad \textit{iff}$$
$$\exists \bar{x}.Q_A(\bar{x}) \wedge Q_{A,R,B}(\bar{x}, \bar{y}) \wedge Q_{A,R,B}(\bar{x}, \bar{z}) \subseteq_S \bigwedge_i \epsilon(y_i, z_i).$$

Example 4.5. Take the transformation T_0 and the schemas S_0 and S_1 in Figure 1. The schema S_1 requires every vaccine to target at least one antigen, in symbols $Vaccine \sqsubseteq \exists targets.Antigen$. This statement is entailed by T_0 and S_0 if and only if the following holds

$$(Vaccine)(x) \subseteq_{S_0} \exists y.(designTarget \cdot crossReacting^*)(x, y).$$ □

For schema elicitation, we use a close correspondence between schemas and $\mathcal{L}_0$ TBoxes. It is sufficient to construct the TBox $\mathcal{T}$ containing all $\mathcal{L}_0$ statements that are entailed by T and S; $\mathcal{T}$ corresponds to the containment-minimal target schema.

Finally, the equivalence of two transformations T_1 and T_2 is essentially the equivalence (modulo S) of the respective queries Q_A

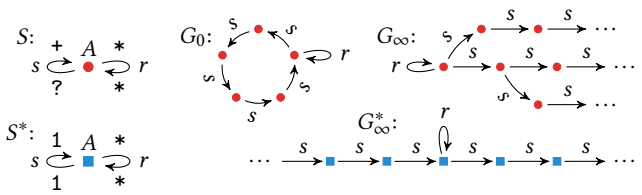

Figure 2: Query containment over finite and infinite graphs.

and $Q_{A,R,B}$ of both transformations. Naturally, query equivalence is reduced to query containment, as usual.

We have shown that type checking, schema elicitation, and equivalence of graph transformations are Turing-reducible in polynomial time to testing containment of UC2RPQs in acyclic UC2RPQs modulo schema. We also show polynomial-time reductions of containment of 2RPQs modulo schema to all above problems of interest. With that, Theorem 4.2 follows from Theorem 5.1.

5 QUERY CONTAINMENT MODULO SCHEMA

The aim of this section is to show the following result.

Theorem 5.1. *Containment of UC2RPQs in acyclic UC2RPQs modulo schema is* EXPTIME*-complete.*

The lower bound can be derived from the EXPTIME-hardness of unrestricted containment of 2RPQs (using only edge labels) modulo very simple TBoxes. The latter is obtained by reduction from another reasoning task (satisfiability of $\mathcal{ALCI}$ TBoxes) and relies on the inner workings of its hardness proof. For completeness, we provide a direct reduction from the acceptance problem for polynomial-space alternating Turing machines. The remainder of this section is devoted to the upper bound. We show it by reduction to unrestricted (finite or infinite) satisfiability of C2RPQs modulo a Horn-$\mathcal{ALCIF}$ TBox, which we discuss in Section 6. The principal technique applied in the reduction is *cycle reversing* [20].

Let S be a schema, P a UC2RPQ, and Q an acyclic UC2RPQ. Without loss of generality we may assume that P and Q are Boolean. The key idea is to pass from finite to possibly infinite graphs, thus making canonical witnesses for non-containment easier to find. However, as Example 5.2 shows, we cannot pass freely from finite to possibly infinite graphs, as this may affect the answer.

Example 5.2. Consider the schema S in Figure 2. Observe that S allows infinite graphs that are essentially infinite trees when restricted to s-edges, e.g. G_∞ in Figure 2. In fact, every infinite graph satisfying S that is connected when restricted to s-edges is an infinite tree. On the other hand, every non-empty finite graph that conforms to S is a collection of disjoint cycles when restricted to s-edges, e.g., G_0 in Figure 2. Clearly, the topology of finite and infinite graphs defined by the schema differs drastically.

Now, take the queries $P = \exists x.r(x,x)$, $Q = \exists x,y.(r \cdot s^+ \cdot r)(x,y)$, and observe that $P \subseteq_S Q$. However, the containment does not hold over infinite graphs: P is satisfied by G_∞ while Q is not. □

The reason why we cannot pass directly to infinite models is that finite graphs conforming to schema S may display certain additional common properties, detectable by queries, but not shared by infinite graphs conforming to S. The cycle *reversing technique* [20] captures

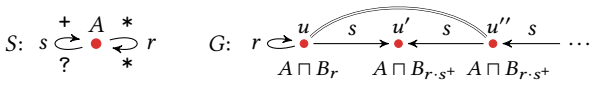

Figure 3: Cycle reversal argument.

these properties in S^* such that

$$P \subseteq_S Q \qquad \textit{iff} \qquad P \subseteq^{\infty}_{S^*} Q$$

where by $\subseteq^{\infty}_{S^*}$ we mean containment over possibly infinite graphs conforming to S^*. However, as the following example shows, we cannot obtain S^* by analysing S alone.

Example 5.3. In Example 5.2 we saw that in a finite graph conforming to S, each node has exactly one incoming and one outgoing s-edge. We can use this observation to tighten the original schema S to the schema S^* (Figure 2). Alas, we still have $P \not\subseteq^{\infty}_{S^*} Q$ because there is an infinite graph G^*_∞ that satisfies P but not Q. □

Instead, we first reduce containment modulo schema to finite satisfiability, fusing the schema S and the query Q into a single Horn-$\mathcal{ALCIF}$ TBox, and then pass from finite to unrestricted satisfiability by applying cycle reversing to the resulting TBox. We follow closely the approach of Ibáñez-García et al. [38], relying crucially on some of their results.

Let $\mathcal{T}$ be a Horn-$\mathcal{ALCIF}$ TBox. A *finmod cycle* is a sequence

$$K_1, R_1, K_2, R_2, \ldots, K_{n-1}, R_{n-1}, K_n$$

where $R_1, \ldots, R_{n-1} \in \Sigma^{\pm}$ and $K_1, \ldots, K_n$ are conjunctions of concept names such that $K_n = K_1$ and

$$\mathcal{T} \models K_i \sqsubseteq \exists R_i.K_{i+1} \quad \text{and} \quad \mathcal{T} \models K_{i+1} \sqsubseteq \exists^{\leq 1} R_i^-.K_i$$

for $1 \leq i < n$. By *reversing* the finmod cycle we mean extending $\mathcal{T}$ with concept inclusions

$$K_{i+1} \sqsubseteq \exists R_i^-.K_i \quad \text{and} \quad K_i \sqsubseteq \exists^{\leq 1} R_i.K_{i+1}$$

for $1 \leq i < n$. The *completion* $\mathcal{T}^*$ of a TBox $\mathcal{T}$ is obtained from $\mathcal{T}$ by exhaustively reversing finmod cycles. The following key result is stated in [38] in terms of sets of ground facts (so-called ABoxes) rather than subgraphs, but our formulation is equivalent.

Theorem 5.4 (Ibáñez-García et al., 2014)**.** *A Horn-$\mathcal{ALCIF}$ TBox $\mathcal{T}$ has a finite model containing a finite subgraph H iff its completion $\mathcal{T}^*$ has a possibly infinite model containing H.*

Example 5.5. Schema S from Example 5.2 is equivalent to TBox $\mathcal{T}_S$ that consists of

$$\top \sqsubseteq A\,, \quad A \sqsubseteq \exists s.A\,, \quad A \sqsubseteq \exists^{\leq 1} s^-.A\,.$$

Non-satisfaction of Q is captured by TBox $\mathcal{T}_{\neg Q}$ that consists of

$$\top \sqsubseteq \forall r.B_r\,, \quad B_r \sqsubseteq \forall s.B_{r\cdot s^+}\,, \quad B_{r\cdot s^+} \sqsubseteq \forall s.B_{r\cdot s^+}\,, \quad B_{r\cdot s^+} \sqsubseteq \forall r.\bot\,.$$

Let $\mathcal{T} = \mathcal{T}_S \cup \mathcal{T}_{\neg Q}$ and observe that $A \sqcap B_{r\cdot s^+}$, s, $A \sqcap B_{r\cdot s^+}$ is a finmod cycle in $\mathcal{T}$. By reversing it, we obtain

$$A \sqcap B_{r\cdot s^+} \sqsubseteq \exists s^-.A \sqcap B_{r\cdot s^+} \quad \text{and} \quad A \sqcap B_{r\cdot s^+} \sqsubseteq \exists^{\leq 1} s.A \sqcap B_{r\cdot s^+}\,.$$

Now, suppose that there exists a (finite or infinite) model G of $\mathcal{T}^*$ that satisfies P (see Figure 3). G must have a node u with $(u,u) \in r^G$. It follows already from $\mathcal{T}$ that $u \in (A \sqcap B_r)^G$ and that u has an s-successor $u' \in (A \sqcap B_{r\cdot s^+})^G$. The statement $A \sqcap B_{r\cdot s^+} \sqsubseteq \exists s^-.A \sqcap B_{r\cdot s^+}$ in $\mathcal{T}^*$ implies that u' has an s^--successor $u'' \in (A \sqcap B_{r\cdot s^+})^G$. As each

node has at most one incoming s-edge, $u = u''$ and $u \in (B_{r\cdot s^+})^G$. But u has an outgoing r-edge, which contradicts the last concept inclusion in $\mathcal{T}_{\neg Q}$. Thus, P is not satisfied in $\mathcal{T}^*$. □

We are now ready to reduce containment modulo schema to unrestricted satisfiability modulo Horn-$\mathcal{ALCIF}$ TBox. Note that the guarantees on the resulting TBox in the statement below are sufficient to conclude Theorem 5.1 using Theorem 6.1.

Theorem 5.6. *Given a UC2RPQ P, an acyclic UC2RPQ Q, and a schema S, one can compute in* EXPTIME *a UC2RPQ $\widehat{P}$ of polynomial size and a Horn-$\mathcal{ALCIF}$ TBox $\mathcal{T}$ using linearly many additional concept names and polynomially many at-most constraints, such that $P \subseteq_S Q$ if and only if $\widehat{P}$ is (unrestrictedly) unsatisfiable modulo $\mathcal{T}$.*

Let us sketch the proof. Let $\mathcal{T}_S$ be the Horn-$\mathcal{ALCIF}$ TBox corresponding to S. Note that apart from the explicit restrictions captured in $\mathcal{T}_S$ the schema S also ensures that only graphs with *exactly* one label per node are considered. To ensure *at most* one label from Γ_S per node, we use the TBox $\widehat{\mathcal{T}_S} = \mathcal{T}_S \cup \{A \sqcap B \sqsubseteq \bot \mid A, B \in \Gamma_S, A \neq B\}$. The concept inclusion $\top \sqsubseteq \bigsqcup \Gamma_S$, expressing that each node has *at least* one label from Γ_S, is not Horn and cannot be used. Instead, we modify the query P. Assuming $\Gamma_S = \{A_1, A_2, \ldots, A_n\}$, we include $(A_1 + A_2 + \cdots + A_n)$ before and after each edge label used in an atom of P. Additionally, to ensure that P uses only labels allowed by S, we substitute in P each label not in $\Gamma_S \cup \Sigma_S^{\pm}$ by $\varnothing$. Letting $\widehat{P}$ be the resulting query, we have

$$P \subseteq_S Q \qquad \textit{iff} \qquad \widehat{P} \subseteq_{\widehat{\mathcal{T}_S}} Q.$$

Because Q is acyclic, by adapting the rolling-up technique [35] one can compute in PTIME a Horn-$\mathcal{ALCIF}$ TBox $\mathcal{T}_{\neg Q}$ over an extended set of concept names $\Gamma_S \cup \Gamma_Q$ such that

$$\widehat{P} \subseteq_{\widehat{\mathcal{T}_S}} Q \qquad \textit{iff} \qquad \widehat{P} \text{ is finitely unsatisfiable modulo } \widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}\,.$$

Since $\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}$ is a Horn-$\mathcal{ALCIF}$ TBox, we can consider its completion $\big(\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}\big)^*$. As UC2RPQs are witnessed by finite subgraphs whenever they are satisfied, we can infer from Theorem 5.4 that $\widehat{P}$ is finitely satisfiable modulo $\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}$ iff $\widehat{P}$ is satisfiable modulo $\big(\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}\big)^*$.

It remains to compute the completion. Reversing cycles does not introduce new concept names, but it may generate exponentially many concept inclusions. Identifying a finmod cycle involves deciding unrestricted entailment of Horn-$\mathcal{ALCIF}$ concept inclusions, which is decidable in EXPTIME [26]. However, since the input TBox might grow to an exponential size as more and more cycles are reversed, it is unlikely that the completion can be computed in EXPTIME for every Horn-$\mathcal{ALCIF}$ TBox. Our key insight is that $\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}$ enjoys a particular property, invariant under reversing cycles, that keeps the complexity under control.

A concept inclusion (CI) of the form $K \sqsubseteq \exists R.K'$ or $K \sqsubseteq \exists^{\leq 1} R.K'$ is *relevant* for a TBox $\mathcal{T}$ if the triple (K, R, K') is satisfiable modulo $\mathcal{T}$; that is, some model G of $\mathcal{T}$ contains nodes u and u' such that $u \in K^G$, $(u, u') \in R^G$, and $u' \in (K')^G$. We say that $\mathcal{T}$ is *S-driven* if for each relevant CI in $\mathcal{T}$ of the form $K \sqsubseteq \exists R.K'$ (resp. $K \sqsubseteq \exists^{\leq 1} R.K'$), $\mathcal{T}$ contains $A \sqsubseteq \exists R.A'$ (resp. $A \sqsubseteq \exists^{\leq 1} R.A'$) for some $A, A' \in \Gamma_S$ such that $A \in K$, $A' \in K'$; here and later we blur the distinction between conjunctions of concept names and sets of labels. Note that $\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}$ is trivially S-driven, as all its existential and at-most constraints are of the form $A \sqsubseteq \exists R.A'$ or $A \sqsubseteq \exists^{\leq 1} R.A'$.

Lemma 5.7. *Every S-driven TBox $\mathcal{T}$ can be simplified in polynomial time so that it contains at most $|\Sigma_S^{\pm}| \cdot |\Gamma_S|^2$ at-most constraints.*

From our results in Section 6 it follows that unrestricted entailment for a Horn-$\mathcal{ALCIF}$ TBox $\mathcal{T}$ with k concept names and ℓ at-most constraints can be solved in time $O\big(\mathrm{poly}(|\mathcal{T}|) \cdot 2^{\mathrm{poly}(k,\ell)}\big)$. Hence, it would suffice to show that by reversing a finmod cycle in an S-driven TBox, we obtain another S-driven TBox. In fact, we prove something weaker, but sufficient to compute the completion in EXPTIME, and conclude that it is S-driven.

Let $K_1, R_1, \ldots, K_{n-1}, R_{n-1}, K_n$ be a finmod cycle in an S-driven Horn-$\mathcal{ALCIF}$ TBox $\mathcal{T}$. Reversing it will extend $\mathcal{T}$ with CIs

$$K_{i+1} \sqsubseteq \exists R_i^-.K_i \quad \text{and} \quad K_i \sqsubseteq \exists^{\leq 1} R_i.K_{i+1}$$

for $1 \leq i < n$. If all triples (K_i, R_i, K_{i+1}) are unsatisfiable wrt $\mathcal{T}$, then all CIs to be added are irrelevant for $\mathcal{T}$ and we are done. Suppose that some (K_i, R_i, K_{i+1}) is satisfiable. Then, in the model for (K_i, R_i, K_{i+1}) we can trace the finmod cycle forward, witnessing each triple. Hence, the whole cycle is satisfiable i.e., all its triples are. Then, we can show that there are unique $A_1, A_2, \ldots, A_n \in \Gamma_S$ such that $A_i \in K_i$ for all $i \leq n$, and $A_1, R_1, \ldots, A_{n-1}, R_{n-1}, A_n$ is a finmod cycle in $\mathcal{T}$. By reversing it, we can add to $\mathcal{T}$ CIs

$$A_{i+1} \sqsubseteq \exists R_i^-.A_i \quad \text{and} \quad A_i \sqsubseteq \exists^{\leq 1} R_i.A_{i+1}$$

for $1 \leq i < n$, which makes the resulting extension S-driven.

Based on the obtained invariant we can compute the completion $\big(\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}\big)^*$ in EXPTIME. By reducing $\big(\widehat{\mathcal{T}_S} \cup \mathcal{T}_{\neg Q}\big)^*$ as described above, we obtain the desired TBox $\mathcal{T}$, thus completing the proof of Theorem 5.6.

6 SATISFIABILITY MODULO TBOX

The last missing piece is to solve the unrestricted satisfiability of C2RPQs modulo Horn-$\mathcal{ALCIF}$. Calvanese et al. show that the problem is in EXPTIME not only for Horn-$\mathcal{ALCIF}$, but even for $\mathcal{ALCIF}$ extended with additional features [16]. This result is not directly applicable, because our reduction produces a TBox of exponential size. The following theorem gives the more precise complexity bounds that we need.

Theorem 6.1. *Unrestricted satisfiability of a C2RPQ p modulo an $\mathcal{ALCIF}$ TBox $\mathcal{T}$ using k concept names and ℓ at-most constraints can be decided in time $O\big(\mathrm{poly}(|\mathcal{T}|) \cdot 2^{\mathrm{poly}(|p|,k,\ell)}\big)$.*

Calvanese et al. solve the problem by first showing a simple model property and then providing an algorithm testing existence of simple models. We rely on the same simple model property, but design a new algorithm with the desired complexity bounds. Yet, rather than diving into the details of the algorithm, we devote most of this section to the simple model property. We do it to show a connection to an elegant graph-theoretical notion that helps to simplify the reasoning considerably, at least for $\mathcal{ALCIF}$. We begin by illustrating how simple models are obtained for queries satisfiable modulo schemas (rather than arbitrary TBoxes).

Example 6.2. Take the schema S in Figure 4 (its two types are represented with a blue square and a red circle), and consider the

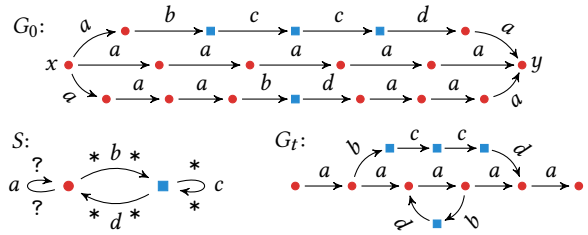

Figure 4: Simple witness for satisfiability.

following satisfiable (cyclic) query

$$p(x,y) = (a \cdot b \cdot c^+ \cdot d \cdot a)(x,y) \wedge (a^*)(x,y) \wedge (a^* \cdot b \cdot d \cdot a^*)(x,y)\,.$$

Since p is satisfiable modulo S, we take any graph conforming to S where p is satisfied, and we choose any 3 paths witnessing each of the regular expressions of p. We construct the initial graph G_0 consisting of the 3 paths joined at their ends: it might look like the one in Figure 4. We observe that S requires every red circle node to have at most one outgoing a-edge and at most one incoming a-edge (to and from a red circle node). The initial graph G_0 violates this requirement and to enforce it we exhaustively merge offending nodes. The final graph G_t is a simple model of p modulo S. □

We formalise simple models using a graph-theoretic notion of sparsity proposed by Lee and Streinu [44]. We say that a connected graph G with n nodes and m edges is *c-sparse* if $m \leq n+c$. (In Lee and Streinu's terminology this corresponds to $(1,-c)$-sparsity.) Being c-sparse is preserved under adding and removing nodes of degree 1. By exhaustively removing nodes of degree 1 from a c-sparse graph G we arrive at single node or a connected c-sparse graph H in which all nodes have degree at least 2. Assuming $c \geq 1$, it is not hard to see that such a graph consists of at most $k = 2c$ distinguished nodes connected by at most $l = 3c$ simple paths disjoint modulo endpoints. We call such a graph a *(k,l)-skeleton*, and we refer to the graph H above as the *skeleton of* G. Thus, a c-sparse graph consists of a $(2c, 3c)$-skeleton and a number of attached trees; by attaching a tree to a graph we mean taking their disjoint union and adding a single edge between the root of the tree and some node of the graph.

For the purpose of the simple model property we need to lift the notion of c-sparsity to infinite graphs. We call a (possibly infinite) graph *c-sparse* if it consists of a finite connected c-sparse graph with finitely many finitely branching trees attached.

Theorem 6.3. *A connected C2RPQ p is satisfiable in a possibly infinite model of an $\mathcal{ALCIF}$ TBox $\mathcal{T}$ iff p is satisfiable in a possibly infinite $|p|$-sparse model of $\mathcal{T}$.*

Proof. Let c be the difference between the number of atoms and the number of variables of p. Because p is connected, $c \geq -1$. By definition, p understood as a graph with variables as nodes and atoms as edges is c-sparse.

We write $H \longrightarrow H'$ to indicate that there is a *homomorphism* from graph H to graph H'; that is, a function h mapping nodes of H to nodes of H' that preserves node labels and the existence of labelled edges between pairs of nodes. Let G be a (possibly infinite) model of p and $\mathcal{T}$. We construct a sequence of finite connected c-sparse graphs of strictly decreasing size

$$G_0 \longrightarrow G_1 \longrightarrow \cdots \longrightarrow G_t \longrightarrow G$$

such that $G_0 \models p$ and the homomorphism from G_t to G is injective over R-successors of every node, for each R.

To construct G_0 let us fix a match of p in G together with a (finite) witnessing path for each atom of p. We construct G_0 as follows. For each variable x of p we include a node v_x whose set of labels is identical to that of the image of x in G under the fixed match. Next, for each atom of p that connects variables x and y we add a simple path connecting x and y such that the sequence of edge labels and sets of node labels read off of this path is identical to that of the witnessing path of this atom in G. This graph can be seen as a specialization of p where each regular expression is replaced by a single concrete word, except that we include full sets of labels of nodes, as they are encountered in the witnessing path in G. It follows immediately that $G_0 \models p$ and that $G_0 \longrightarrow G$. To see that G_0 is c-sparse one can eliminate the internal nodes of the connecting paths one by one, until a graph isomorphic to p remains.

We define the remaining graphs G_i inductively, maintaining an additional invariant $G_i \longrightarrow G$. Suppose we already have G_i together with a homomorphism $h_i \colon G_i \longrightarrow G$ for some $i \geq 0$. If h_i is injective over R-successors of each node of G_i, we are done. If not, there are two different R-successors u_1 and u_2 of a node v in G_i that are mapped to the same node u' in G. It follows that u_1 and u_2 have the same sets of labels types. We let G_{i+1} be the graph obtained from G_i by merging u_1 and u_2 into a single node u. We include an R'-edge between u and each R'-successor of u_1 or u_2. This decreases the number of nodes by one, and the number or edges by at least one. It follows that G_{i+1} is c-sparse and $G_i \longrightarrow G_{i+1} \longrightarrow G$.

Because the sizes of graphs G_i are strictly decreasing, at some point we will arrive at a graph G_t such that the homomorphism from G_t to G is injective over R-successors.

The graph G_t clearly satisfies p. It also satisfies all concept inclusions in $\mathcal{T}$ of the forms $K \sqsubseteq A_1 \sqcup A_2 \sqcup \cdots \sqcup A_n$, $K \sqsubseteq \bot$, $K \sqsubseteq \forall R.K'$, $K \sqsubseteq \nexists R.K'$, and $K \sqsubseteq \exists^{\leq 1} R.K'$, because h_i is injective over R-successors and $G \models \mathcal{T}$. On the other hand, G_t is not guaranteed to satisfy concept inclusions of the form $K \sqsubseteq \exists R.K'$ in $\mathcal{T}$. In order to fix it, we exhaustively (ad infinitum) perform the following: whenever a node v in G_t is missing an R-successor with some set of labels, we add it and map it to some such R-successor u' of the image of v in G (u' exists because $G \models \mathcal{T}$). As $c \leq |p|$, the resulting (typically infinite) graph $\widehat{G}$ is $|p|$-sparse, and it satisfies p and $\mathcal{T}$. □

The connectedness assumption in Theorem 6.3 is not restrictive, because a witnessing graph for p can be obtained by taking the disjoint union of witnesses for its connected components. Hence, it remains to decide for a given connected p if there exists a $|p|$-sparse graph G that satisfies p and $\mathcal{T}$. To get a finer control of the effect different parameters of the input have on the complexity, we side-step two-way alternating tree automata (2ATA) applied by Calvanese et al. and develop a more direct algorithm.

Observe that if p is satisfied in a $|p|$-sparse graph G, then G contains a $(4|p|, 5|p|)$-skeleton H', extending the skeleton of G, such that all variables of p are mapped to distinguished nodes of H'. Indeed, H' can be obtained by iteratively extending the skeleton of G. Suppose that some variable is mapped to a node v that is not

yet a distinguished node of H'. If v already belongs to H', then it is an internal node in a path between two distinguished nodes; we then split the path in two, turning v into a distinguished node. If v does not belong to H', then it belongs to a tree attached to H' at a node u. If u is not a distinguished node of H', we turn it into one, as above. Then, we add v to H' as a distinguished node, including the path between u and v into H' as well. As we start from a $(2|p|, 3|p|)$-skeleton and add at most two distinguished nodes and two paths for each variable of p, we end up with a $(4|p|, 5|p|)$-skeleton.

Thus, the algorithm can guess a $(4|p|, 5|p|)$-skeleton H' with each path represented by a single symbolic edge and check that it can be completed to a suitable graph G by materializing symbolic edges into paths and attaching finitely many finitely branching trees in such a way that G is a model of $\mathcal{T}$ and there is a match of p in G that maps variables of p to distinguished nodes of H'. This can be done within the required time bounds by means of a procedure that can be seen as a variant of type elimination or an emptiness test for an implicitly represented nondeterministic tree automaton.

7 DISCUSSION

Summary. In this paper we have studied several static analysis problems for graph transformations defined with Datalog-like rules that use acyclic C2RPQs. When the source schema is given, we studied the *equivalence* problem of two given transformations, and the problem of *target schema elicitation* for a given transformation. If the output schema is also given, we have studied the problem of *type checking*. We have shown that the above problems can be reduced to containment of C2RPQs in acyclic UC2RPQs modulo schema, a problem that we have reduced to the unrestricted (finite or infinite) satisfiabilty of a C2RPQ modulo Horn-$\mathcal{ALCIF}$ TBox using cycle reversing. For the latter problem we have presented an algorithm with sufficiently good complexity to accommodate the exponential blow-up introduced by cycle reversing, thus allowing to solve in EXPTIME all problems of interest. We have also shown matching lower bounds by reducing query containment modulo schema to each of the static analysis problems.

Finite containment modulo Horn-$\mathcal{ALCIF}$ TBox. In the course of the proof of the upper bound for containment modulo schema, we essentially solved (finite) containment modulo Horn-$\mathcal{ALCIF}$ TBox. Indeed, while the EXPTIME upper bound relies on the special shape of the TBox expressing the schema, the method can be applied directly to any Horn-$\mathcal{ALCIF}$ TBox, at the cost of an exponential increase in complexity. Thus, we immediately get that finite containment of UC2RPQs in acyclic UC2RPQs modulo Horn-$\mathcal{ALCIF}$ TBoxes can be solved in 2EXPTIME. To the best of our knowledge this is the first result on finite containment of C2RPQs in the context of description logics. A related problem of finite entailment has been studied for various logics [27–29, 31], but while for conjunctive queries the solutions carry over to finite containment, for C(2)RPQs these logics are too weak to allow this. Unrestricted containment of C2RPQs modulo $\mathcal{ALCIF}$ TBoxes is known to be in 2EXPTIME [16], but passing from unrestricted to finite structures is typically challenging for such problems. For example, finite entailment of CRPQs for a fundamental description logic $\mathcal{ALC}$ has been solved only recently [31], 15 years after the unrestricted version [14].

Extending queries. It is straightforward to extend our methods to two-way *nested regular expressions* (NREs) [52]. We also intend to investigate introducing *negation* in filter expressions of NREs. Eliminating the acyclicity assumption, on the other hand, is problematic. Containment of arbitrary C2RPQs is EXPSPACE-complete [15], and we have shown that it reduces to our problems of interest for transformation rules with cyclic queries. Hence, extending our EXPTIME upper bounds to transformations allowing cyclic C2RPQs is highly unlikely. In fact, even establishing decidability would be hard. For acyclic queries we could use the rolling-up technique to reduce containment to satisfiability, which allowed us to apply the cycle reversing technique and pass from finite to unrestricted models. When cyclic queries are allowed, the rolling-up technique is inapplicable and we are left with containment of C2RPQs modulo constraints, which is a major open problem, not only for constraints expressed in description logics. The only positive results we are aware of do not go significantly beyond CQs extended with a binary reachability relation [23].

Extending schemas. Extending the schema formalism with disjunction is also challenging: the corresponding description logic would not be Horn any more and the transition to unrestricted models via cycle reversing would not be possible. Supporting multiple labels on nodes would not be a trivial extension either: we rely on the single label per node assumption in the reduction of the problems of interest to containment of UC2RPQs in *acyclic* UC2RPQs, and in the EXPTIME upper bound. Supporting more general cardinality constraints, on the other hand, should be possible, but it might affect the complexity upper bounds.

Extending the data model. It is straightforward to encode data values in our graph model, for instance, by using dedicated node labels to designate *literal* nodes whose identifiers are their data values. Then, one can apply methods similar to type checking to verify that transformations are well-behaved, and in particular, do not attempt to construct literal nodes from non-literal ones. However, the full consequences of allowing literal values in definitions of transformation rules need to be thoroughly investigated.

Finally, we have considered equivalence of transformations based on equality of results but one could also consider a variant based on isomorphism of results. This would be an entirely different problem, probably much harder.

ACKNOWLEDGMENTS

This work was supported by Poland's National Science Centre grant 2018/30/E/ST6/00042. We would like to thank Sebastian Maneth, Mikaël Monet, Bruno Guillon, and Yazmin Ibáñez-García for their comments and discussions. For the purposes of open access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Serge Abiteboul and Richard Hull. 1987. IFO: A Formal Semantic Database Model. *ACM Trans. Database Syst.* 12, 4 (Nov. 1987), 525–565. https://doi.org/10.1145/32204.32205

[2] Serge Abiteboul and Paris C. Kanellakis. 1998. Object Identity as a Query Language Primitive. *J. ACM* 45, 5 (Sept. 1998), 798–842. https://doi.org/10.1145/290179.290182

[3] Renzo Angles. 2018. The Property Graph Database Model. In *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018 (CEUR Workshop Proceedings, Vol. 2100)*. CEUR-WS.org. http://ceur-ws.org/Vol-2100/paper26.pdf

[4] Renzo Angles and Claudio Gutierrez. 2008. Survey of graph database models. *Comput. Surveys* 40, 1 (Feb. 2008), 1–39. https://doi.org/10.1145/1322432.1322433

[5] Marcelo Arenas, Jorge Pérez, Juan Reutter, and Cristian Riveros. 2013. The language of plain SO-tgds: Composition, inversion and structural properties. *J. Comput. System Sci.* 79 (Sept. 2013). https://doi.org/10.1016/j.jcss.2013.01.002

[6] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. 2017. *An Introduction to Description Logic*. Cambridge University Press.

[7] Pablo Barceló, Jorge Pérez, and Juan L. Reutter. 2013. Schema mappings and data exchange for graph databases. In *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*. https://doi.org/10.1145/2448496.2448520

[8] Mikolaj Bojanczyk, Leszek Aleksander Kolodziejczyk, and Filip Murlak. 2013. Solutions in XML data exchange. *J. Comput. Syst. Sci.* 79 (Sept. 2013). https://doi.org/10.1016/j.jcss.2013.01.004

[9] Iovka Boneva, Benoit Groz, Jan Hidders, Filip Murlak, and Sławek Staworko. 2023. *Static Analysis of Graph Database Transformations*. Technical Report. https://hal.science/hal-03937274

[10] Iovka Boneva, Sławek Staworko, and Jose Lozano. 2020. Consistency and Certain Answers in Relational to RDF Data Exchange with Shape Constraints. In *New Trends in Databases and Information Systems*, Vol. 1259. 97–107. https://doi.org/10.1007/978-3-030-54623-6_9

[11] Angela Bonifati, Peter Furniss, Alastair Green, Russ Harmer, Eugenia Oshurko, and Hannes Voigt. 2019. Schema Validation and Evolution for Graph Databases. In *Conceptual Modeling*. 448–456. https://doi.org/10.1007/978-3-030-33223-5_37

[12] Peter Buneman, Mary Fernandez, and Dan Suciu. 2000. UnQL: A Query Language and Algebra for Semistructured Data Based on Structural Recursion. *The VLDB Journal* 9 (2000), 76–110. https://doi.org/10.1007/s007780050084

[13] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y Vardi. 2011. Simplifying schema mappings. In *Proceedings of the 14th International Conference on Database Theory*. 114–125. https://doi.org/10.1145/1938551.1938568

[14] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. 2007. Answering Regular Path Queries in Expressive Description Logics: An Automata-Theoretic Approach. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*. AAAI Press, 391–396. http://www.aaai.org/Library/AAAI/2007/aaai07-061.php

[15] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000*. 176–185.

[16] Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. 2011. Containment of Regular Path Queries under Description Logic Constraints. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. 805–812. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-141

[17] Peter P. Chen. 1975. The Entity-Relationship Model: Toward a Unified View of Data. In *Proceedings of the International Conference on Very Large Data Bases, September 22-24, 1975, Framingham, Massachusetts, USA*. 173. https://doi.org/10.1145/1282480.1282492

[18] Cristina Civili, Jose Mora, Riccardo Rosati, Marco Ruzzi, and Valerio Santarelli. 2016. Semantic Analysis of R2RML Mappings for Ontology-Based Data Access. In *Web Reasoning and Rule Systems*. 25–38. https://doi.org/10.1007/978-3-319-45276-0_3

[19] Julien Corman, Juan L. Reutter, and Ognjen Savkovic. 2018. Semantics and Validation of Recursive SHACL. In *The Semantic Web – ISWC 2018*. 318–336. https://doi.org/10.1007/978-3-030-00671-6_19

[20] Stavros S. Cosmadakis, Paris C. Kanellakis, and Moshe Y. Vardi. 1990. Polynomial-Time Implication Problems for Unary Inclusion Dependencies. *J. ACM* 37, 1 (1990), 15–46. https://doi.org/10.1145/78935.78937

[21] Bruno Courcelle. 1994. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science* 126 (1994). https://doi.org/10.1016/0304-3975(94)90268-2

[22] Richard Cyganiak, Seema Sundara, and Souripriya Das. 2012. *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. W3C. https://www.w3.org/TR/2012/REC-r2rml-20120927/.

[23] Alin Deutsch and Val Tannen. 2002. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *Database Programming Languages*. 21–39. https://doi.org/10.1007/3-540-46093-4_2

[24] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336 (2005). https://doi.org/10.1016/j.tcs.2004.10.033

[25] Ingo Feinerer, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. 2015. On the undecidability of the equivalence of second-order tuple generating dependencies. *Information Systems* 48 (2015). https://doi.org/10.1016/j.is.2014.09.003

[26] Giuseppe De Giacomo and Maurizio Lenzerini. 1996. TBox and ABox Reasoning in Expressive Description Logics. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*. 316–327.

[27] Tomasz Gogacz, Víctor Gutiérrez-Basulto, Albert Gutowski, Yazmín Ibáñez-García, and Filip Murlak. 2020. On Finite Entailment of Non-Local Queries in Description Logics. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*. 424–433. https://doi.org/10.24963/kr.2020/43

[28] Tomasz Gogacz, Víctor Gutiérrez-Basulto, Yazmín Ibáñez-García, Jean Christoph Jung, and Filip Murlak. 2019. On Finite and Unrestricted Query Entailment beyond SQ with Number Restrictions on Transitive Roles. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. 1719–1725. https://doi.org/10.24963/ijcai.2019/238

[29] Tomasz Gogacz, Yazmín Angélica Ibáñez-García, and Filip Murlak. 2018. Finite Query Answering in Expressive Description Logics with Transitive Roles. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR*. 369–378.

[30] W3C: RDF Working Group. 2004. Resource Description Framework. https://www.w3.org/RDF/. Accessed: 2022, June 03.

[31] Victor Gutiérrez-Basulto, Albert Gutowski, Yazmin Ibáñez-Garcia, and Filip Murlak. 2022. Finite Entailment of UCRPQs over ALC Ontologies. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR*. 184–194. https://doi.org/10.24963/kr.2022/19

[32] Marc Gyssens, Jan Paredaens, Jan van den Bussche, and Dirk van Gucht. 1994. A graph-oriented object database model. *IEEE Transactions on Knowledge and Data Engineering* 6, 4 (1994), 572–586. https://doi.org/10.1109/69.298174

[33] Jan Hidders. 2003. Typing Graph-Manipulation Operations. In *Database Theory — ICDT 2003*. 391–406. https://doi.org/10.1007/3-540-36285-1_26

[34] HL7.org. 2019. FHIR Mapping Language. https://hl7.org/fhir/mapping-language.html. Accessed: 2022-05-25.

[35] Ian Horrocks and Sergio Tessaris. 2000. A Conjunctive Query Language for Description Logic Aboxes. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. 399–404.

[36] Richard Hull and Masatoshi Yoshikawa. 1990. ILOG: Declarative Creation and Manipulation of Object Identifiers. In *Proceedings of the 16th International Conference on Very Large Data Bases*. 455–468. http://www.vldb.org/conf/1990/P455.PDF

[37] Richard Hull and Masatoshi Yoshikawa. 1991. On the Equivalence of Database Restructurings Involving Object Identifiers (Extended Abstract). In *Proceedings of the Tenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 328–340. https://doi.org/10.1145/113413.113443

[38] Yazmín Angélica Ibáñez-García, Carsten Lutz, and Thomas Schneider. 2014. Finite Model Reasoning in Horn Description Logics. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*. 288–297.

[39] Kazuhiro Inaba, Soichiro Hidaka, Zhenjiang Hu, Hiroyuki Kato, and Keisuke Nakano. 2011. Graph-Transformation Verification Using Monadic Second-Order Logic. In *Proceedings of the 13th International ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming*. 17–28. https://doi.org/10.1145/2003476.2003482

[40] Michael Kay. 2021. *XSL Transformations (XSLT) Version 2.0 (Second Edition)*. W3C Recommendation. W3C. https://www.w3.org/TR/2021/REC-xslt20-20210330/.

[41] Michael Kifer and Georg Lausen. 1989. F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme. *SIGMOD Rec.* 18, 2 (June 1989), 134–146. https://doi.org/10.1145/66926.66939

[42] Phokion G. Kolaitis, Reinhard Pichler, Emanuel Sallinger, and Vadim Savenkov. 2020. On the Language of Nested Tuple Generating Dependencies. *ACM Trans. Database Syst.* 45, 2 (2020), 8:1–8:59. https://doi.org/10.1145/3369554

[43] Gabriel M. Kuper and Moshe Y. Vardi. 1993. The Logical Data Model. *ACM Trans. Database Syst.* 18, 3 (Sept. 1993), 379–413. https://doi.org/10.1145/155271.155274

[44] Audrey Lee and Ileana Streinu. 2008. Pebble game algorithms and sparse graphs. *Discret. Math.* 308, 8 (2008), 1425–1437. https://doi.org/10.1016/j.disc.2007.07.104

[45] Mark Levene and Alexandra Poulovassilis. 1990. The hypernode model and its associated query language. In *Proceedings of the 5th Jerusalem Conference on Information Technology, 1990. 'Next Decade in Information Technology'*. 520–530. https://doi.org/10.1109/JCIT.1990.128324

[46] David Maier. 1986. A Logic for Objects. In *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*. 6 – 26.

[47] Sebastian Maneth, Alexandru Berlea, Thomas Perst, and Helmut Seidl. 2005. XML type checking with macro tree transducers. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. https://doi.org/10.1145/1065167.1065203

[48] Wim Martens and Frank Neven. 2007. Frontiers of tractability for typechecking simple XML transformations. *J. Comput. System Sci.* 73 (2007). https://doi.org/10.1016/j.jcss.2006.10.005

[49] Wim Martens, Frank Neven, and Marc Gyssens. 2008. Typechecking top-down XML transformations: Fixed input or output schemas. *Information and Computation* 206, 7 (2008), 806–827. https://doi.org/10.1016/j.ic.2008.01.002

[50] Tova Milo, Dan Suciu, and Victor Vianu. 2003. Typechecking for XML transformers. *J. Comput. System Sci.* 66 (2003). https://doi.org/10.1016/S0022-0000(02)00030-2

[51] Jan Paredaens, Peter Peelman, and Letizia Tanca. 1995. G-Log: a graph-based query language. *IEEE Transactions on Knowledge and Data Engineering* 7, 3 (June 1995), 436–453. https://doi.org/10.1109/69.390249

[52] Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. 2010. nSPARQL: A navigational language for RDF. *J. Web Semant.* 8, 4 (2010), 255–270. https://doi.org/10.1016/j.websem.2010.01.002

[53] Eric Prud'hommeaux, Harold R. Solbrig, and Guoqian Jiang. 2017. ShEx, RDF and FHIR. In *Summit on Clinical Research Informatics, CRI 2017, San Francisco, CA, USA, March 27-30, 2017*.

[54] Grzegorz Rozenberg (Ed.). 1997. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific.

[55] Juan F. Sequeda. 2013. On the Semantics of R2RML and its Relationship with the Direct Mapping. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013 (CEUR Workshop Proceedings, Vol. 1035)*. 193–196. https://ceur-ws.org/Vol-1035/iswc2013_poster_4.pdf

[56] Josh Spiegel, Michael Dyck, and Jonathan Robie. 2017. *XQuery 3.1: An XML Query Language*. W3C Recommendation. W3C. https://www.w3.org/TR/2017/REC-xquery-31-20170321/.

[57] Sławek. Staworko, Iovka Boneva, Jose Emilio Labra Gayo, Samuel Hym, Eric G. Prud'hommeaux, and Harold Solbrig. 2015. Complexity and Expressiveness of ShEx for RDF. In *International Conference on Database Theory (ICDT)*. 195–211. https://doi.org/10.4230/LIPIcs.ICDT.2015.195

[58] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *International Conference on World Wide Web (WWW)*. 697–706. https://doi.org/10.1145/1242572.1242667