Querying the Unary Negation Fragment with Regular Path Expressions

Jean Christoph Jung
University of Bremen, Bremen, Germany
jeanjung@informatik.uni-bremen.de

Carsten Lutz
University of Bremen, Bremen, Germany
clu@informatik.uni-bremen.de

Mauricio Martel
University of Bremen, Bremen, Germany
martel@informatik.uni-bremen.de

Thomas Schneider
University of Bremen, Bremen, Germany
ts@informatik.uni-bremen.de

Abstract

The unary negation fragment of first-order logic (UNFO) has recently been proposed as a generalization of modal logic that shares many of its good computational and model-theoretic properties. It is attractive from the perspective of database theory because it can express conjunctive queries (CQs) and ontologies formulated in many description logics (DLs). Both are relevant for ontology-mediated querying and, in fact, CQ evaluation under UNFO ontologies (and thus also under DL ontologies) can be 'expressed' in UNFO as a satisfiability problem. In this paper, we consider the natural extension of UNFO with regular expressions on binary relations. The resulting logic UNFO^{reg} can express (unions of) conjunctive two-way regular path queries (C2RPQs) and ontologies formulated in DLs that include transitive roles and regular expressions on roles. Our main results are that evaluating C2RPQs under UNFO^{reg} ontologies is decidable, 2ExpTime-complete in combined complexity, and coNP-complete in data complexity, and that satisfiability in UNFO^{reg} is 2ExpTime-complete, thus not harder than in UNFO.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Query Answering, Regular Path Queries, Decidable Fragments of First-Order Logic, Computational Complexity

Digital Object Identifier 10.4230/LIPIcs.ICDT.2018.15

Funding A full version of this paper with an appendix that contains detailed proofs is available at `http://www.informatik.uni-bremen.de/tdki/research/papers.html`

Acknowledgements ERC Consolidator Grant 647289 CODA and DFG grant SCHN 1234/3.

1 Introduction

In ontology-mediated querying, queries against incomplete and heterogeneous data are supported by an ontology that provides domain knowledge and assigns a semantics to the data [15, 19, 37, 41]. The ontologies are often formulated in a specialized language such as a description logic [4, 5] or an existential rule language [6, 7, 22, 33] while the actual query is typically a conjunctive query (CQ) or a mild extension thereof such as a union of CQs

21st International Conference on Database Theory (ICDT 2018).
Editors: Benny Kimelfeld and Yael Amsterdamer; Article No. 15; pp. 15:1–15:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(UCQ). However, it can also be useful to consider more expressive decidable fragments of first-order logic (FO) as an ontology language as this serves to explore the limits of the ontology-mediated querying approach, to provide maximum expressive power for ontology formulation, and to put ontology-mediated querying into a more general logical perspective. Notably, this has been done in [8, 9, 19], where the guarded fragment (GF), the unary negation fragment (UNFO), and the guarded negation fragment (GN) of FO have been used as ontology languages. These fragments originate from the attempt to explain the good computational behaviour of modal and description logics and to extend their expressive power in a natural way. While GF and UNFO are orthogonal in expressive power, GN subsumes both of these fragments [9] and all of them subsume many common modal and description logics. It is an important result that, for all these fragments, ontology-mediated querying with UCQs remains decidable and that the complexity stays within the expected, namely 2ExpTime combined and coNP data.

From the perspective of database theory, it is an attractive property of both UNFO and GN (but not of GF) that they can express CQs and UCQs. In ontology-mediated querying, this allows to 'express' the evaluation of ontology-mediated queries in terms of satisfiability in a natural way. It is easiest to state this for Boolean queries: if $(\mathcal{O}, \Sigma, q)$ is an ontology-mediated query (OMQ) where $\mathcal{O}$ is an ontology, Σ a set of predicate symbols (that is, relation names) that may occur in the data, and q a UCQ, and D is a Σ-database, then $D \models (\mathcal{O}, \Sigma, q)$ iff $\mathcal{O} \wedge D \wedge \neg q$ is unsatisfiable. When $\mathcal{O}$ is formulated in UNFO or in GN, then so is $\mathcal{O} \wedge D \wedge \neg q$. What is more, the containment of OMQs can also be 'expressed' as a satisfiability problem in the natural case where both OMQs contain the same ontology and Σ is the set of all predicates symbols; from now on, we generally mean this case when speaking of OMQ containment. But also beyond ontology-mediated querying, we believe that the ability to express UCQs makes UNFO and GN attractive as an expressive logical backdrop for database theory.

In this paper, we study the natural extension $\text{UNFO}^{\mathsf{reg}}$ of UNFO with regular path expressions on binary relations. The resulting logic has the attractive property that it allows to express regular path queries [29] and conjunctive two-way regular path queries (C2RPQs) [25] as well as unions thereof (UC2RPQs). Such queries play a central role in the area of graph databases [2, 10] and they have also received considerable attention in ontology-mediated querying [12, 17, 18, 26, 27, 28, 40]. An additional reason to consider $\text{UNFO}^{\mathsf{reg}}$ is provided by the observation that transitive roles are an important feature of many common description logics (a role is a binary relation), but that transitive roles cannot be expressed in UNFO. In $\text{UNFO}^{\mathsf{reg}}$, even transitive closure of roles and regular expressions on roles are expressible, two features that are provided by several expressive description logics [3, 24]. As a concrete example, every ontology formulated in $\mathcal{ALCI}^{\mathsf{reg}}$, the extension of the common description logic $\mathcal{ALCI}$ with regular expressions on roles [44], can be expressed in $\text{UNFO}^{\mathsf{reg}}$ and thus the evaluation of ontology-mediated queries $(\mathcal{O}, \Sigma, q)$ where $\mathcal{O}$ is formulated in $\mathcal{ALC}^{\mathsf{reg}}$ and q is a UC2RPQ can be 'expressed' as a satisfiability problem in $\text{UNFO}^{\mathsf{reg}}$; of course, the same is true when $\mathcal{O}$ is formulated in $\text{UNFO}^{\mathsf{reg}}$ itself. We remark that transitive roles cannot be expressed in GF and GNF either, and that adding transitive relations to GF without losing decidability requires to impose rather strong syntactical restrictions [45], especially so in an ontology-mediated querying context [34]. Adding transitive relations to GNF has, to the best of our knowledge, not yet been studied.

The main problem that we are interested in is evaluating OMQs in which the ontology is formulated in $\text{UNFO}^{\mathsf{reg}}$ and the actual query is a UC2RPQ. We show that this problem is decidable, 2ExpTime-complete in combined complexity and coNP-complete in data complexity. We further consider the OMQ containment problem and show that it is 2ExpTime-complete as well. We additionally show that the complexity of model checking in UNFO$^{\mathsf{reg}}$ is the same as in UNFO, namely complete for $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$.

As explained above, both OMQ evaluation and OMQ containment can be reduced to satisfiability in polynomial time. For studying the combined complexity of the former and the complexity of the latter, we thus concentrate on the satisfiability problem and prove a 2ExpTime upper bound. Note that the addition of regular expressions does thus not increase the complexity of this problem as satisfiability in UNFO is also 2ExpTime-complete [46] and that the lower bound holds already when the arity of predicates is bounded by two, as a consequence of the results in [39]. Our proof proceeds by first showing that every satisfiable UNFO$^{\mathsf{reg}}$ formula φ has a model whose treewidth is bounded by the size of φ, then establishing a characterization of the satisfaction of C2RPQs (that occur as a building block in φ) in such models in terms of certain witness trees, and finally showing that this infrastructure gives rise to a decision procedure based on two-way alternating tree automata. This 'direct approach' is in contrast to the reduction to satisfiability in the μ-calculus used for UNFO in [46] which seems unwieldy in the presence of regular path expressions. Note in particular that an important reason for the relative simplicity of the reduction in [46] is that there is always a model of bounded treewidth in which any two bags overlap in at most one element; this is no longer true in UNFO$^{\mathsf{reg}}$. To establish the coNP upper bound on data complexity, we first observe that it suffices to consider a database satisfiability problem (given a database D, is there a model of the fixed UNFO$^{\mathsf{reg}}$ sentence φ that extends D?) and then establish a certain kind of decoration of D as a witness for D being a positive instance, in a way such that witnesses can be guessed and verified in polynomial time.

Related work. For general background on ontology-mediated querying, we refer to [15, 19, 37, 41] and the references therein. OMQ containment was considered in [11, 13, 14, 21]. UNFO was introduced and studied by ten Cate and Segoufin in [46] and it was considered as an ontology language for OMQs in [19]. Regular path queries, C2RPQs, and variations thereof emerge from graph databases, see the surveys [2, 10] and references therein. We use C2RPQs that admit nesting via node tests, as considered in [16], see also [20]. Sometimes, this is referred to as 'nested' C2RPQs. There are several further extensions of C2RPQs that are not considered in this paper. We still mention two of them. A more powerful form of nesting is obtained by allowing to use C2RPQs with two answer variables in place of binary predicates in regular expressions, giving rise to regular queries [42]. Another expressive extension of C2RPQs is defined by monadically defined queries, which implement a certain 'flag and check' paradigm [43]. OMQs in which the actual query is some form of regular path queries are considered in [12, 17, 18, 26, 27, 28, 40]. As discussed in more detail later, UNFO$^{\mathsf{reg}}$ is also related (but orthogonal in expressive power) to propositional dynamic logic with intersection and converse (ICPDL) [31] and to UNFO extended with fixed points [46].

2 Preliminaries

We assume that a countably infinite supply of predicate symbols of each arity is available. In the *unary negation fragment of first-order logic extended with regular path expressions* (UNFO$^{\mathsf{reg}}$), formulas φ are formed according to the following grammar:

$$\begin{array}{lcl} \varphi & ::= & P(\mathbf{x}) \mid E(x,y) \mid x = y \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \neg \varphi(x) \\ E & ::= & R \mid R^- \mid E \cup E \mid E \cdot E \mid E^* \mid \varphi(x)? \end{array}$$

where P ranges over predicate symbols, R over binary predicate symbols, and, in the $\neg\varphi(x)$ clause, φ has no free variables besides (possibly) x. Expressions E formed according to the second line are called *(regular) path expressions* and expressions $\varphi(x)?$ according to the last clause in that line are called *tests*. Tests are similar to the test operator in propositional dynamic logic (PDL) [30] and to node tests in XPath [32] and in some versions of regular path queries [16, 20]. When we write $\varphi(\mathbf{x})$, we generally mean that the free variables of φ are among $\mathbf{x}$, but not all variables from $\mathbf{x}$ need actually be free in φ. For a UNFO$^{\mathsf{reg}}$ formula $\varphi(x)$, we use $\forall x\varphi$ to abbreviate $\neg\exists x\neg\varphi(x)$.

▶ **Example 1.** The following are UNFO$^{\mathsf{reg}}$ formulas: $\forall x\big(\exists yR(x,y) \wedge \neg(R \cdot R^*)(x,x)\big)$ and $\exists y\big(R^*(x,y) \wedge S^*(x,y)\big)$.

A *structure* $\mathfrak{A}$ takes the form $(A, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \dots)$ where A is a non-empty set called the *domain* and R_i is an n_i-ary relation over A if R_i is a predicate symbol of arity n_i.Whenever convenient, we use $\mathsf{dom}(\mathfrak{A})$ to refer to A. Every path expression E is interpreted as a binary relation $E^{\mathfrak{A}}$ over A: $R^{\mathfrak{A}}$ is part of $\mathfrak{A}$, $(R^-)^{\mathfrak{A}}$ is the converse of $R^{\mathfrak{A}}$, $(E_1 \cup E_2)^{\mathfrak{A}} = E_1^{\mathfrak{A}} \cup E_2^{\mathfrak{A}}$, $(E_1 \circ E_2)^{\mathfrak{A}} = E_1^{\mathfrak{A}} \circ E_2^{\mathfrak{A}}$, $(E^*)^{\mathfrak{A}}$ is the reflexive-transitive closure of $E^{\mathfrak{A}}$, and $(\varphi(x)?)^{\mathfrak{A}} = \{(a,a) \mid \mathfrak{A} \models \varphi(a)\}$. UNFO$^{\mathsf{reg}}$ formulas are then interpreted under the standard first-order semantics with path expressions being treated in the same way as binary predicates. A UNFO$^{\mathsf{reg}}$ sentence $\varphi(\mathbf{x})$ is *satisfiable* if there is a structure $\mathfrak{A}$ such that $\mathfrak{A} \models \varphi$. Such an $\mathfrak{A}$ is called a *model* of $\varphi(\mathbf{x})$.

▶ **Example 2.** Reconsider the UNFO$^{\mathsf{reg}}$ formulas from Example 1. It can be verified that the first sentence is satisfiable, but not in a finite model. Thus, in contrast to UNFO (and to propositional dynamic logic), UNFO$^{\mathsf{reg}}$ lacks the finite model property. The second sentence expresses a property that cannot be expressed in UNFO extended with fixed points, as studied in [46], which can formally be shown using UN-bisimulations, also defined in [46]. In fact, UNFO$^{\mathsf{reg}}$ and UNFO with fixed points are orthogonal in expressive power. Another related logic is ICPDL, that is, PDL extended with intersection and converse [31]. This logic, too, is orthogonal in expressive power to UNFO$^{\mathsf{reg}}$. For example, the existence of a 4-clique can be expressed as a UNFO sentence, but not in ICPDL since every satisfiable ICPDL formula is satisfiable in a structure of tree width two.

The expressive power of UNFO$^{\mathsf{reg}}$ is closely related to that of conjunctive 2-way regular path queries. A *database* D is a finite structure such that for every $a \in \mathsf{dom}(D)$, there is an $\mathbf{a} \subseteq \mathsf{dom}(D)$ and a predicate symbol P such that $a \in \mathbf{a} \in P^D$. Since a database is a syntactic object, we refer to the elements of $\mathsf{dom}(D)$ as *constants* whereas we speak about *elements* in the context of (semantic) structures. A *conjunctive 2-way regular path query (C2RPQ)* is a formula of the form $q(\mathbf{x}) = \exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$ where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of *atoms* of the form $R(\mathbf{z})$ and $E(z_1, z_2)$, R a predicate symbol and E a two-way regular path query, that is, an expression formed according to the second line of the syntax definition of UNFO$^{\mathsf{reg}}$, but allowing only formulas $\varphi(x)$ that are C2RPQs in tests. The variables $\mathbf{x}$ are the *answer variables* of $q(\mathbf{x})$ and $q(\mathbf{x})$ is *Boolean* if $\mathbf{x} = \emptyset$. A *union of C2RPQs (UC2RPQ)* is a disjunction of C2RPQs that all have the same answer variables. A *conjunctive query (CQ)* is a C2RPQ that does not use atoms of the form $E(z_1, z_2)$. The *answers* to a UC2RPQ $q(\mathbf{x})$ on a database D, denoted $\mathsf{ans}(q, D)$, are defined in the standard way, see for example [42]. Note that every UC2RPQ is a UNFO$^{\mathsf{reg}}$ formula.

▶ **Example 3.** Consider the following database about family relationships, using binary predicates Child and Spouse, and written as a set of facts.

$$\begin{aligned} D \;=\; \{\, & \mathsf{Child}(\text{Nívea}, \text{Clara}),\ \mathsf{Child}(\text{Clara}, \text{Blanca}),\ \mathsf{Child}(\text{Blanca}, \text{Alba}), \\ & \mathsf{Spouse}(\text{Nívea}, \text{Severo}),\ \mathsf{Spouse}(\text{Esteban}, \text{Clara}) \quad \} \end{aligned}$$

The following C2RPQ asks for all pairs (x, y) such that x is an ancestor of y in a line of only married ancestors (using the shorthand $R^+ = R \cdot R^*$).

$$q(x,y) = (m(z)? \cdot \mathsf{Child})^+(x,y) \quad \text{where} \quad m(z) = \exists z' \,(\mathsf{Spouse} \cup \mathsf{Spouse}^-)(z,z')$$

We have $\mathsf{ans}(q, D) = \{(\mathsf{Nívea}, \mathsf{Clara}), (\mathsf{Nívea}, \mathsf{Blanca}), (\mathsf{Clara}, \mathsf{Blanca})\}$.

Let $q(\mathbf{x}) = \exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$ be a C2RPQ. We use $\mathsf{var}(q)$ to denote the variables that occur in q outside of tests, that is, $\mathbf{x} \cup \mathbf{y}$. We do not distinguish between $q(\mathbf{x})$ and the set of all atoms in φ, writing e.g. $R(x, y, z) \in q(\mathbf{x})$ to mean that $R(x, y, z)$ is an atom in φ. For simplicity, we treat an atom $E(x, x)$ in a C2RPQ $q(\mathbf{x})$ where E is the test $\varphi(y)?$ as an atom of the form $\varphi(x)$; that is, w.l.o.g. we use tests not only in path expressions but also directly as atoms of a C2RPQ. A C2RPQ $q(\mathbf{x})$ can be viewed as a finite hypergraph in the expected way, that is, every atom $R(\mathbf{z})$ and $E(z_1, z_2)$ is viewed as a hyperedge. We say that $q(\mathbf{x})$ is *connected* if the Gaifman graph of this hypergraph is connected. It is interesting to observe that foundational problems concerning UC2RPQs can be phrased as (un)satisfiability problems in $\text{UNFO}^{\mathsf{reg}}$.

▶ **Example 4.**

1. The problem whether a Boolean UC2RPQ $q()$ evaluates to true on a database D (i.e., whether the empty tuple is in $\mathsf{ans}(q, D)$) corresponds to the unsatisfiability of $\varphi_D() \wedge \neg q()$ where $\varphi_D()$ is D viewed as a Boolean CQ in the obvious way.
2. The problem whether a Boolean UC2RPQ $q_1()$ is contained in a Boolean UC2RPQ $q_2()$ (defined in the usual way) corresponds to the unsatisfiability of $q_1() \wedge \neg q_2()$.

Both reductions extend to the case of non-Boolean queries by simulating answer variables using fresh unary predicates, see the proof of Lemma 6 below.

An *ontology-mediated query (OMQ)* is a triple $(\mathcal{O}, \Sigma, q)$ where $\mathcal{O}$ is a logical sentence called the *ontology*, Σ is a set of predicate symbols called the *data signature*, and q is a query. In this paper, we shall primarily be interested in the case where $\mathcal{O}$ is an $\text{UNFO}^{\mathsf{reg}}$ sentence and q is a UC2RPQ. We use $(\text{UNFO}^{\mathsf{reg}}, \text{UC2RPQ})$ to denote the set of OMQs of this form and similarly for other ontology languages and query languages. Let $Q = (\mathcal{O}, \Sigma, q)$ be from $(\text{UNFO}^{\mathsf{reg}}, \text{UC2RPQ})$ and D a database that uses only symbols from Σ. We call $\mathbf{a} \subseteq \mathsf{dom}(D)$ a *certain answer to* Q *on* D if $\mathbf{a} \in \mathsf{ans}(q, \mathfrak{A})$ for every structure $\mathfrak{A}$ that extends D and is a model of $\mathcal{O}$, where $\mathfrak{A}$ *extends* D if $\mathsf{dom}(D) \subseteq \mathsf{dom}(\mathfrak{A})$ and $P^D \subseteq P^{\mathfrak{A}}$ for all predicate symbols P. Note that this semantics embodies a 'standard names assumption', that is, constants in D are interpreted as themselves. The set of all certain answers to Q on D is denoted $\mathsf{cert}(Q, D)$. We say that Q is *Boolean* if q is. For a Boolean OMQ Q, we write $D \models Q$ to indicate that Q is true on D, meaning that the empty tuple is in $\mathsf{cert}(Q, D)$.

▶ **Example 5.** Consider the OMQ $Q = (\mathcal{O}, \Sigma, q')$ based on an extension of the C2RPQ q from Example 3, where $\mathcal{O}$ defines a single mother as an unmarried woman who has a child, using additional unary predicates Female and $\mathsf{SingleMother}$, and q' has an additional conjunct requiring that y is a single mother, that is:

$$\begin{aligned}
\mathcal{O} &= \forall x \,\big(\mathsf{SingleMother}(x) \leftrightarrow \mathsf{Female}(x) \wedge \mathsf{Single}(x) \wedge \exists y\, \mathsf{Child}(x,y)\big)\\
q'(x,y) &= q(x,y) \wedge \mathsf{SingleMother}(y)\\
\Sigma &= \{\mathsf{Child}, \mathsf{Spouse}, \mathsf{Female}, \mathsf{Single}\}
\end{aligned}$$

Note that $\mathcal{O}$ is equivalent to a $\text{UNFO}^{\mathsf{reg}}$ (even plain UNFO) formula obtained by eliminating $\leftrightarrow$ in the usual way. Let $D' = D \cup \{\mathsf{Female}(\mathsf{Blanca}), \mathsf{Single}(\mathsf{Blanca})\}$, where D is the database from Example 3. Then $\mathsf{cert}(Q, D') = \{(\mathsf{Nívea}, \mathsf{Blanca}), (\mathsf{Clara}, \mathsf{Blanca})\}$, but $\mathsf{cert}(Q, D) = \emptyset$.

OMQ evaluation in (UNFO$^{\mathsf{reg}}$, UC2RPQ) is the problem to decide, given an OMQ Q from (UNFO$^{\mathsf{reg}}$, UC2RPQ), a database D, and an $\mathbf{a} \subseteq \mathsf{dom}(D)$, whether $\mathbf{a} \in \mathsf{cert}(Q, D)$. This is a relevant problem since ontologies formulated in many logics used as ontology languages can be translated into an equivalent UNFO$^{\mathsf{reg}}$ sentence in polynomial time. In particular, this is the case for the basic description logics $\mathcal{ALC}$ and $\mathcal{ALCI}$ [19] and for their extensions with transitive closure of roles [3] and with regular expressions over roles [24]. For any of these logics $\mathcal{L}$, this of course also yields a polynomial time reduction of OMQ evaluation in ($\mathcal{L}$, UC2RPQ) to OMQ evaluation in (UNFO$^{\mathsf{reg}}$, UC2RPQ). Even UNFO itself has occasionally been considered as an ontology language [19].

For the rather common extension of the description logic $\mathcal{ALC}$ with transitive roles [5], an equivalence preserving translation of ontologies into UNFO$^{\mathsf{reg}}$ sentences is not possible since UNFO$^{\mathsf{reg}}$ cannot enforce that a binary predicate is transitive. However, a transitive role can be simulated using the transitive closure of a binary predicate R (and never using R without transitive closure). In this way, one still obtains the desired polynomial time reduction of OMQ evaluation. The same reduction can be applied even to UNFO$^{\mathsf{reg}}$ extended with transitive relations. We use UNFO$^{\mathsf{reg}}_{\mathsf{trans}}$ to denote the extension of UNFO$^{\mathsf{reg}}$ where sentences take the form $\varphi_{\mathsf{trans}} \wedge \varphi$ with φ_{trans} a conjunction of atoms of the form $\mathsf{trans}(R)$, R a binary predicate symbol, and φ a UNFO$^{\mathsf{reg}}$ sentence. An atom $\mathsf{trans}(R)$ is satisfied in a structure $\mathfrak{A}$ if $R^{\mathfrak{A}}$ is transitive.

Evaluation of Boolean OMQs in (UNFO$^{\mathsf{reg}}_{\mathsf{trans}}$, UC2RPQ) reduces in polynomial time to satisfiability in UNFO$^{\mathsf{reg}}_{\mathsf{trans}}$ since $D \models (\mathcal{O}, \Sigma, q)$ iff $\varphi_D() \wedge \mathcal{O} \wedge \neg q()$ is unsatisfiable. The reduction can be extended to non-Boolean queries by simulating answer variables using fresh unary predicates. Because of this observation, in the main body of the paper we concentrate on deciding satisfiability rather than OMQ evaluation.

▶ **Lemma 6.** *Lemma OMQ evaluation in* (UNFO$^{\mathsf{reg}}_{\mathsf{trans}}$, UC2RPQ) *reduces in polynomial time to satisfiability in* UNFO$^{\mathsf{reg}}$*, and so does satisfiability in* UNFO$^{\mathsf{reg}}_{\mathsf{trans}}$.

Together with Theorem 14 it thus follows that UNFO can be extended with transitive relations without losing decidability or affecting the complexity of satisfiability and of OMQ evaluation. This is in contrast to the guarded fragment, where in both cases decidability can only be obtained by adopting additional syntactic restrictions. While for satisfiability it suffices to assume that transitive relations are only used in guard positions, even stronger restrictions are necessary for OMQ evaluation [35, 45, 34].

There are also other interesting reasoning problems that can be reduced to satisfiability in UNFO$^{\mathsf{reg}}$. Here we consider OMQ containment, leaving out transitive roles for simplicity. Let $Q_1 = (\mathcal{O}, \Sigma_{\mathsf{full}}, q_1)$ and $Q_2 = (\mathcal{O}, \Sigma_{\mathsf{full}}, q_2)$ be OMQs from (UNFO$^{\mathsf{reg}}$, UC2RPQ) with the same number of answer variables and where Σ_{full} is the *full* data signature, that is, the set of all predicate symbols. We say that Q_1 *is contained in* Q_2 and write $Q_1 \subseteq Q_2$ if for every database D, $\mathsf{cert}(Q_1, D) \subseteq \mathsf{cert}(Q_2, D)$. We observe that OMQ containment can also be reduced to satisfiability in polynomial time.

▶ **Lemma 7.** *OMQ containment in* (UNFO$^{\mathsf{reg}}$, UC2RPQ) *reduces in polynomial time to satisfiability in* UNFO$^{\mathsf{reg}}$.

There are also versions of OMQ containment that admit different ontologies in the two involved OMQs and more restricted data signatures in place of Σ_{full} [11, 13, 14, 21]. These are computationally harder and a polynomial time reduction to satisfiability cannot be expected. In fact, it follows from results in [21] that this more general form of OMQ containment is 2NExpTime-hard already when the ontologies are formulated in the description logic $\mathcal{ALCI}$, a fragment of UNFO, and when the actual queries are CQs. Decidability remains an open problem. We remark that when the actual queries in OMQs are CQs, then OMQ containment under the full data signature can be reduced to query evaluation in a straightforward way, essentially by viewing the query from the left-hand OMQ as a database. In the presence of regular path queries, however, this does not seem to be easily possible.

We next introduce a normal form for UNFO^{reg} sentences, similar but not identical to the normal form used for UNFO in [46]. For a set $\mathcal{L}$ of UNFO^{reg} formulas with one free variable, a *C2RPQ extended with $\mathcal{L}$-formulas* is a C2RPQ in which all tests $\varphi(x)$? in atoms $E(z_1, z_2)$ have been replaced with tests $\psi(x)$?, $\psi(x)$ a formula from $\mathcal{L}$. The set of *normal* UNFO^{reg} *formulas* is the smallest set of formulas such that

1. every connected C2RPQ with exactly one free variable, extended with normal UNFO^{reg} formulas, is a normal UNFO^{reg} formula;
2. if $\varphi(x)$ and $\psi(x)$ are normal UNFO^{reg} formulas, then $\neg\varphi(x)$, $\varphi(x) \vee \psi(x)$, and $\exists x\, \varphi(x)$ are normal UNFO^{reg} formulas.

Observe that Item 1 serves as an induction start since every connected C2RPQ without tests (and with one free variable) is a normal UNFO^{reg} formula. Note that normal formulas are closed under conjunction in the sense that the conjunction of normal formulas $\varphi_1(x)$ and $\varphi_2(x)$ is a C2RPQ extended with normal UNFO^{reg} formulas and thus a normal formula. Thus, unary disjunction could be eliminated, but for our purposes it is more convenient to keep it. We note in passing that using this normal form, it is easy to observe that UNFO^{reg} has the same expressive power as C2RPQs that admit both tests and negated tests.

The *width* of a normal UNFO^{reg} formula is the maximal number of variables in a C2RPQ that occurs in it (not counting the variables that occur in the C2RPQ only inside tests). The *atom width* is defined analogously, but referring to the number of atoms instead of the number of variables. In the context of normal UNFO^{reg} formulas, for brevity we speak of C2RPQs when meaning C2RPQs extended with normal UNFO^{reg} formulas. The *size* of a UNFO^{reg} formula is the number of symbols needed to write it, with variable symbols and predicate symbols being counted as a single symbol.

▶ **Lemma 8.** *Every UNFO^{reg} sentence φ can be transformed into an equivalent normal UNFO^{reg} sentence φ' in single exponential time. Moreover, the width and the atom width of φ' are at most polynomial in the size of φ and the path expressions that occur in φ' are exactly those in φ.*

In the following sections, we replace atoms $E(z_1, z_2)$ in the C2RPQs that occur in a normal UNFO^{reg} formula with atoms of the form $\mathcal{A}(z_1, z_2)$ where $\mathcal{A}$ is a *nondeterministic automaton on finite words (NFA)* over a suitable alphabet; we call such atoms *NFA atoms*. Formally, an NFA is a tuple $(Q, \Sigma, \Delta, q_0, F)$ where Q is a finite set of states, Σ a finite alphabet, $\Delta \subseteq Q \times \Sigma \times Q$ a transition relation, $q_0 \in Q$ an initial state and $F \subseteq Q$ a set of final states. When deciding the satisfiability of a UNFO^{reg} sentence φ_0, we will generally take Σ to be $\{R, R^- \mid R \text{ a binary predicate in } \varphi_0\} \cup \{\varphi(x)? \mid \varphi(x)? \text{ a test in } \varphi_0\}$. Clearly, all path expressions in φ_0 are regular expressions over this alphabet. Since every regular expression can be converted into an equivalent NFA in polynomial time, we can thus w.l.o.g. assume the NFA-based presentation. Let $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ be an NFA. Then we use $\mathcal{A}[F/F']$ to denote the NFA obtained from $\mathcal{A}$ by replacing F with $F' \subseteq Q$ and $\mathcal{A}[q_0/q]$ for the NFA obtained from $\mathcal{A}$ by replacing q_0 with $q \in Q$. For a structure $\mathfrak{A}$, an NFA $\mathcal{A}$, and $a, b \in A$, we write $\mathfrak{A} \models \mathcal{A}(a, b)$ if there are $a_1, \ldots, a_n \in A$ and a word $w \in L(\mathcal{A})$ of length $n - 1$ such that $a = a_1$, $a_n = b$, $(a_i, a_{i+1}) \in R^{\mathfrak{A}}$ if the ith symbol in w is R, $(a_{i+1}, a_i) \in R^{\mathfrak{A}}$ if the ith symbol in w is R^-, and $a_i = a_{i+1}$ and $\mathfrak{A} \models \varphi(a_i)$ if the ith symbol in w is $\varphi(x)$?. This gives a semantics to NFA atoms. The *size* of a normal UNFO^{reg} formula with NFA atoms is defined in the same way as the size of a UNFO formula, where every NFA $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ contributes the cardinality of Q plus the cardinality of Δ plus the cardinality of F.

3 Tree-like Structures and Witness Trees

We give a characterization of satisfiability in UNFO^{reg} that is tailored towards implementation by tree automata. In particular, we show that every satisfiable UNFO^{reg} formula φ has a model whose treewidth is bounded by the width of φ, introduce a representation of such models in terms of labeled trees, and characterize the satisfaction of C2RPQs in models represented in this way in terms of tree-shaped witnesses. To simplify the technical development, in this section and the subsequent one we disallow predicates of arity zero. Note that an atom $P()$ can be simulated by the formula $\exists x\, P(x)$, so this assumption is w.l.o.g. We work with normal UNFO^{reg} sentences throughout the section.

A *(directed) tree* is a prefix-closed subset $T \subseteq (\mathbb{N} \setminus \{0\})^*$. A node $w \in T$ is a *successor* of $v \in T$ and v is a *predecessor* of w if $w = v \cdot i$ for some $i \in \mathbb{N}$. Moreover, w is a *neighbor* of v if it is a successor or predecessor of v. A *tree-like structure* is a pair (T, bag) where T is a tree and bag a function that assigns to every $w \in T$ a finite structure $\mathsf{bag}(w)$ such that

$$\text{the set of nodes } \{w \in T \mid a \in \mathsf{dom}(w)\} \text{ is connected in } T\text{, for each } a \in \bigcup_{w \in T} \mathsf{dom}(w)$$

where, here and in the remainder of the paper, $\mathsf{dom}(w)$ is a shorthand for $\mathsf{dom}(\mathsf{bag}(w))$. The *width* of (T, bag) is the maximum domain size of structures that occur in the range of bag. Its *outdegree* is the outdegree of T. A tree-like structure (T, bag) defines the associated structure $\mathfrak{A}_{(T,\mathsf{bag})}$ which is the (non-disjoint) union of all structures $\mathsf{bag}(w)$, $w \in T$. We use $\mathsf{dom}(T, \mathsf{bag})$ as a shorthand for $\mathsf{dom}(\mathfrak{A}_{(T,\mathsf{bag})})$. As witnessed by its representation (T, bag), the treewidth of the structure $\mathfrak{A}_{(T,\mathsf{bag})}$ is bounded by the maximum cardinality of $\mathsf{dom}(\mathsf{bag}(w))$, $w \in T$. We will show that every satisfiable UNFO^{reg} sentence φ_0 is satisfiable in a tree-like structure whose width is bounded by the width of φ_0. In UNFO, it suffices to consider structures of this form in which bags overlap in at most one element; this is not the case in UNFO^{reg}.

Let φ_0 be a normal UNFO^{reg} sentence. We use $\mathsf{sub}(\varphi_0)$ to denote the subformulas of φ_0 with at most one free variable, and where the free variable is renamed to x. Then $\mathsf{cl}(\varphi_0)$ denotes the smallest set of normal UNFO^{reg} formulas that contains $\mathsf{sub}(\varphi_0)$ and is closed under single negation. A *1-type for* φ_0 is a subset $t \subseteq \mathsf{cl}(\varphi_0)$ that satisfies the following conditions:

1. $\varphi \in t$ iff $\neg\varphi \notin t$ for all $\neg\varphi \in \mathsf{cl}(\varphi_0)$;
2. $\varphi \vee \psi \in t$ iff $\varphi \in t$ or $\psi \in t$ for all $\varphi \vee \psi \in \mathsf{cl}(\varphi_0)$.

We use $\mathsf{TP}(\varphi_0)$ to denote the set of all 1-types for φ_0.

A *type decorated tree-like structure for* φ_0 is a triple (T, bag, τ) with (T, bag) a tree-like structure such that only predicates from φ_0 occur in the range of bag and $\tau : \mathsf{dom}(T, \mathsf{bag}) \to \mathsf{TP}(\varphi_0)$. Let (T, bag, τ) be such a structure, $\mathcal{A}$ an NFA, and $a, b \in \mathsf{dom}(T, \mathsf{bag})$. We write $\mathfrak{A}_{(T,\mathsf{bag})}, \tau \models \mathcal{A}(a, b)$ if $\mathfrak{A}_{(T,\mathsf{bag})} \models \mathcal{A}(a, b)$ with the semantics of tests reinterpreted: instead of demanding that $\mathfrak{A} \models \varphi(a')$ for a test $\varphi_0(x)?$ to hold at an element a', we now require that $\varphi \in \tau(a')$. Let $\varphi(x) = \exists \mathbf{y}\, \psi(x, \mathbf{y})$ be a C2RPQ and $a \in \mathsf{dom}(T, \mathsf{bag})$. A *homomorphism from* $\varphi(x)$ *to* (T, bag, τ) is a function $h : \{x\} \cup \mathbf{y} \to \mathsf{dom}(T, \mathsf{bag})$ such that the following conditions are satisfied:

- $h(\mathbf{x}) \in R^{\mathfrak{A}_{(T,\mathsf{bag})}}$ for each $R(\mathbf{x}) \in \varphi(x)$;
- $\mathfrak{A}_{(T,\mathsf{bag})}, \tau \models \mathcal{A}(h(y), h(z))$ for each $\mathcal{A}(y, z) \in \varphi(x)$.

A type decorated tree-like structure (T, bag, τ) for φ_0 is *proper* if:

1. for all $\exists x\, \varphi(x) \in \mathsf{cl}(\varphi_0)$, $\exists x\, \varphi(x) \in \tau(a)$ iff there is a $b \in \mathsf{dom}(T, \mathsf{bag})$ with $\varphi(x) \in \tau(b)$;
2. for all C2RPQs $\varphi(x) \in \mathsf{cl}(\varphi_0)$, $\varphi(x) \in \tau(a)$ iff there is a homomorphism h from $\varphi(x)$ to (T, bag, τ) such that $h(x) = a$.

The following lemma establishes proper type decorated tree-like structures for φ_0 as witnesses for the satisfiability of φ_0. The proof of the 'only if' direction is via an unraveling procedure that constructs a type decorated tree-like structure in a top-down manner, introducing fresh bags to satisfy C2RPQs and to implement a step-by-step chase of paths that witness satisfaction of NFA atoms in C2RPQs.

▶ **Lemma 9.** *A normal* $\mathrm{UNFO}^{\mathsf{reg}}$ *sentence* φ_0 *of size* n *and width* m *is satisfiable iff there is a proper type decorated tree-like structure* (T, bag, τ) *for* φ_0 *of width at most* m *and outdegree at most* $n^2 + n$ *such that* $\varphi_0 \in \tau(a)$ *for some* a.

As the next step, we take a closer look at Point 2 of properness, that is, we characterize carefully the existence of a homomorphism h from $\varphi(x)$ to (T, bag, τ) such that $h(x) = a$ in a way that is tailored towards implementation by tree automata. This gives rise to the notion of a witness tree below. We start with introducing the notions of subdivisions and splittings which shall help us to take care of the fact that the homomorphic image of a query $q(\mathbf{x})$ may be spread over several bags of a tree-like structure, and in fact this might even be the case for a single NFA atom.

An *instantiated C2RPQ* is a C2RPQ in which all free variables have been replaced with constants. We write $\varphi(\mathbf{a})$ to indicate that the constants in the instantiated C2RPQ are exactly $\mathbf{a}$. When working with instantiated C2RPQs, we drop existential quantifiers, assuming that all variables are implicitly existentially quantified. For brevity, we often omit the word 'instantiated' and only speak of C2RPQs. We speak of *terms* to mean both variables and constants, and we denote terms with t.

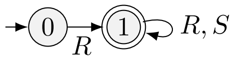
Let $\varphi(\mathbf{a})$ be a connected C2RPQ, Δ be a domain, and $s \geq 1$. A (Δ, s)-*subdivision of* of an atom $\mathcal{A}(t, t') \in \varphi(\mathbf{a})$ is a set of atoms

$$\mathcal{A}[F/\{q_1\}](t, b_1), \mathcal{A}[q_0/q_1, F/\{q_2\}](b_1, b_2), \ldots, \mathcal{A}[q_0/q_{k-1}, F/\{q_k\}](b_{k-1}, b_k), \mathcal{A}[q_0/q_k](b_k, t')$$

where $q_1, \ldots, q_k$ are states of $\mathcal{A}$, $k \leq s$, and $b_1, \ldots, b_k$ are constants from Δ. A C2RPQ $\psi(\mathbf{a}')$ is a (Δ, s)-*subdivision* of $\varphi(\mathbf{a})$ if it is obtained from $\varphi(\mathbf{a})$ by replacing zero or more NFA atoms with (Δ, s)-subdivisions. Let $\psi(\mathbf{a}')$ be a (Δ, s)-subdivision of $\varphi(\mathbf{a})$. A *splitting* of $\psi(\mathbf{a}')$ is a sequence $\psi_0(\mathbf{a}_0), \ldots, \psi_\ell(\mathbf{a}_\ell)$, $\ell \geq 0$, of C2RPQs that is a partition of $\psi(\mathbf{a}')$ (viewed as a set of atoms) where we also allow the special case that $\psi_0(\mathbf{a}_0)$ is empty (and thus $\psi_1(\mathbf{a}_1), \ldots, \psi_\ell(\mathbf{a}_\ell)$ is the actual partition). We require that the following conditions are satisfied:

1. $\psi_1(\mathbf{a}_1), \ldots, \psi_\ell(\mathbf{a}_\ell)$ are connected;
2. $\mathsf{var}(\psi_i(\mathbf{a}_i)) \cap \mathsf{var}(\psi_j(\mathbf{a}_j)) \subseteq \mathsf{var}(\psi_0(\mathbf{a}_0))$ for $1 \leq i < j \leq \ell$;
3. each of $\psi_1(\mathbf{a}_1), \ldots, \psi_\ell(\mathbf{a}_\ell)$ contains at most one atom from each subdivision of an atom in $\varphi(\mathbf{a})$.

Intuitively, the $\vartheta_0(\mathbf{a})$ component of a splitting is the part of $\psi(\mathbf{a}')$ that maps into a bag that we are currently focussing on while the other components are pushed to neighboring bags.

▶ **Example 10.** Consider $q(a) = \{\mathcal{A}(a, y), T(a, z), Q(a, y, z)\}$ with $\mathcal{A} =$ 

. Let $\Delta = \{a, b, c\}$ and $\mathcal{A}_{ij} = \mathcal{A}[0/i, F/j]$. An example for a $(\Delta, 2)$-subdivision of $\mathcal{A}(a, y)$ is $\{\mathcal{A}_{01}(a, b), \mathcal{A}_{11}(b, b), \mathcal{A}_{11}(b, y)\}$, which yields the following $(\Delta, 2)$-subdivision of $\varphi(a)$:

$\psi(a,b) = \{\mathcal{A}_{01}(a,b), \mathcal{A}_{11}(b,b), \mathcal{A}_{11}(b,y), T(a,z), Q(a,y,z)\}$. $\psi(a,b)$ admits a splitting into ψ_0, ψ_1 as follows: $\psi_0(a,b) = \{\mathcal{A}_{01}(a,b), \mathcal{A}_{11}(b,b), T(a,z)\}$ and $\psi_1(a,b) = \{\mathcal{A}_{11}(b,y), Q(y,z,a)\}$.

The *query closure* $\mathsf{qcl}(\varphi_0, \Delta, s)$ is defined as the smallest set such that the following conditions are satisfied:

- if $\varphi(x) \in \mathsf{cl}(\varphi_0)$ is a C2RPQ and $a \in \Delta$, then $\varphi(a) \in \mathsf{qcl}(\varphi_0, \Delta, s)$;
- if $\varphi(\mathbf{a}) \in \mathsf{qcl}(\varphi_0, \Delta, s)$, $\psi(\mathbf{a}')$ is a (Δ, s)-subdivision of $\varphi(\mathbf{a})$, $\psi_0(\mathbf{a}_0), \ldots, \psi_\ell(\mathbf{a}_\ell)$ is a splitting of $\psi(\mathbf{a}')$, $1 \leq i \leq \ell$, and $\psi_i'(\mathbf{a}_i')$ is obtained from $\psi_i(\mathbf{a}_i)$ by consistently replacing zero or more variables with constants from Δ, then $\psi_i'(\mathbf{a}_i') \in \mathsf{qcl}(\varphi_0, \Delta, s)$.

▶ **Lemma 11.** *The cardinality of $\mathsf{qcl}(\varphi_0, \Delta, s)$ is bounded by $p \cdot (a^2 d^m)^{m'}$, where p is the number of C2RPQs in φ_0, a the maximal number of states in an NFA in φ_0, d the cardinality of Δ, m the width of φ_0, and m' the atom width of φ_0.*

We are almost ready to define witness trees. The following notion of a homomorphism is more local than the ones used so far as it only concerns a single bag rather than the entire tree-like structure. Let (T, bag, τ) be a type decorated tree-like structure, $w \in T$, $\mathcal{A}$ an NFA, and $a, b \in \mathsf{dom}(w)$. We write $\mathsf{bag}(w), \tau \models \mathcal{A}(a,b)$ if $\mathsf{bag}(w) \models \mathcal{A}(a,b)$ with the semantics of tests reinterpreted: instead of demanding $\mathsf{bag}(w) \models \varphi(a')$ for a test $\varphi(x)?$ to hold at an element a', we now require that $\varphi(x) \in \tau(a')$. Let $\varphi(\mathbf{a})$ be a C2RPQ. A *homomorphism from $\varphi(\mathbf{a})$ to $\mathsf{bag}(w)$ given τ* is a function $h : \mathbf{a} \cup \mathsf{var}(\varphi) \to \mathsf{dom}(w)$ such that the following conditions are satisfied:

- $h(a) = a$ for each $a \in \mathbf{a}$;
- $h(\mathbf{t}) \in R^{\mathsf{bag}(w)}$ for each $R(\mathbf{t}) \in \varphi(\mathbf{a})$;
- $\mathsf{bag}(w), \tau \models \mathcal{A}(h(t), h(t'))$ for each $\mathcal{A}(t, t') \in \varphi(a)$.

Let n be the size of φ_0, $a \in \mathsf{dom}(T, \mathsf{bag})$, and $\varphi(x) \in \mathsf{cl}(\varphi_0)$ a C2RPQ. A *witness tree for $\varphi(a)$ in (T, bag, τ)* is a finite labeled tree (W, σ) with $\sigma : W \to T \times \mathsf{qcl}(\varphi_0, \mathsf{dom}(T, \mathsf{bag}), n^2)$ such that the root is labeled with $\sigma(\varepsilon) = (w, \varphi(a))$ for some $w \in T$ with $a \in \mathsf{dom}(w)$ and the following conditions are satisfied for all $u \in W$:

$(*)$ if $\sigma(u) = (w, \psi(\mathbf{a}))$, then there is a $(\mathsf{dom}(w), n^2)$-subdivision $\vartheta'(\mathbf{a})$ of $\psi(\mathbf{a})$, a splitting $\vartheta_0(\mathbf{a}_0), \ldots, \vartheta_\ell(\mathbf{a}_\ell)$ of $\vartheta'(\mathbf{a})$, a homomorphism h from $\vartheta_0(\mathbf{a}_0)$ to $\mathsf{bag}(w)$ given τ, and successors $u_1, \ldots, u_\ell$ of u such that $\sigma(u_i) = (w_i, \vartheta_i'(\mathbf{a}_i'))$ for $1 \leq i \leq \ell$, where each w_i is a neighbor of w in T with $\mathbf{a}_i' \subseteq \mathsf{dom}(w_i)$ and $\vartheta_i'(\mathbf{a}_i')$ is obtained from $\vartheta_i(\mathbf{a}_i)$ by replacing each variable x in the domain of h with the constant $h(x)$.

Informally, a witness tree decomposes a homomorphism h from $\varphi(x)$ to $\mathfrak{A}_{T,\mathsf{bag}}$ into local 'chunks', each of which concerns only a single bag. In particular, the splitting $\vartheta_0(\mathbf{a}_0), \ldots, \vartheta_k(\mathbf{a}_k)$ in $(*)$ breaks the current C2RPQ down into components that are satisfied in different parts of the tree-like structure.We need to first subdivide since satisfaction of NFA atoms is witnessed by an entire path, and this path can pass through the current node several times. Fortunately, the number of points introduced in a subdivision can be bounded: we can w.l.o.g. choose a shortest path and such a path can pass through w at most once for each element in $\mathsf{dom}(w)$ and each state of the automaton $\mathcal{A}$, thus we need at most n^2 points in subdivisions.

▶ **Lemma 12.** *Let (T, bag, τ) be a type decorated tree-like structure, $\varphi(x) \in \mathsf{cl}(\varphi_0)$ a C2RPQ, and $a \in \mathsf{dom}(T, \mathsf{bag})$. Then there is a homomorphism h from $\varphi(x)$ to (T, bag, τ) with $h(x) = a$ iff there is a witness tree for $\varphi(a)$ in (T, bag, τ).*

4 Automata-Based Decision Procedure

We now reduce satisfiability of $\text{UNFO}^{\mathsf{reg}}$ sentences to the nonemptiness problem of two-way alternating tree automata. We start with recalling this automata model and discuss the encoding of tree-like structures as an input to automata.

Two-way alternating tree automata. A tree is *k-ary* if each node has *exactly k* successors. As a convention, we set $w \cdot 0 = w$ and $wc \cdot (-1) = w$, leave $\varepsilon \cdot (-1)$ undefined, and for any $k \in \mathbb{N}$, set $[k] = \{-1, 0, \ldots, k\}$. Let Σ be a finite alphabet. A *Σ-labeled tree* is a pair (T, L) with T a tree and $L : T \to \Sigma$ a node labeling function.

An *alternating 2-way tree automaton (2ATA)* over Σ-labeled k-ary trees is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ where Q is a finite set of *states*, $q_0 \in Q$ is an *initial state*, δ is the *transition function*, and F is the *(parity) acceptance condition*, that is, a finite sequence $G_1, \ldots, G_k$ with $G_1 \subseteq G_2 \subseteq \ldots \subseteq G_k = Q$. The transition function maps a state q and an input letter $a \in \Sigma$ to a positive Boolean formula over the constants true and false, and variables from $[k] \times Q$. The semantics is given in terms of runs in the appendix of the long version. As usual, $L(\mathcal{A})$ denotes the set of trees accepted by $\mathcal{A}$. The *nonemptiness problem* for 2ATAs is the problem to decide, given a 2ATA $\mathcal{A}$, whether $L(\mathcal{A})$ is nonempty. It can be solved in time single exponential in the number of states and the number of sets in the parity condition, and linear in the size of the transition function [47].

Encoding of tree-like structures. Let φ_0 be a normal $\text{UNFO}^{\mathsf{reg}}$ sentence whose satisfiability we want to decide. By Lemma 9, this corresponds to deciding the existence of a proper type decorated tree-like structure for φ_0 (of certain dimensions) and thus our aim is to build a 2ATA $\mathcal{A}$ such that $L(\mathcal{A}) \neq \emptyset$ if and only if there is such a structure. 2ATAs cannot run directly on tree-like structures because the labeling of the underlying trees is not finite: we have already shown that $\text{UNFO}^{\mathsf{reg}}$ does not have the finite model property and thus it might be necessary that infinitely many elements occur in the bags. We therefor use an appropriate encoding that 'reuses' element names so that we can make do with finitely many element names overall, similar to what has been done, for example, in [36, 1].

Let $R_1, \ldots, R_\ell$ be the predicate symbols that occur in φ_0 and let m be the width of φ_0. Fix a finite set Δ with $2m$ elements and define Σ to be the set of all pairs (bag, τ) such that $\mathsf{bag} = (A, R_1^{\mathsf{bag}}, \ldots, R_\ell^{\mathsf{bag}})$ is a structure that satisfies $A \subseteq \Delta$ and $|A| \leq m$, and $\tau : A \to \mathsf{TP}(\varphi_0)$ is a map that assigns a 1-type to every element in bag.

Let (T, L) be a Σ-labeled tree. For convenience, we use bag_w to refer to the first component of $L(w)$ and τ_w to refer to the second component, that is, $L(w) = (\mathsf{bag}_w, \tau_w)$. Moreover, dom_w is shorthand for $\mathsf{dom}(\mathsf{bag}_w)$. For an element $d \in \Delta$, we say that $v, w \in T$ are *d-equivalent* if $d \in \mathsf{dom}_u$ for all u on the unique shortest path from v to w. Informally, this means that d represents the same element in bag_v and in bag_w. In case that $d \in \mathsf{dom}_w$, we use $[w]_d$ to denote the set of all v that are d-equivalent to w. We say that (T, L) is *type consistent* if, for all $d \in \Delta$ and all d-equivalent $v, w \in T$, $\tau_v(d) = \tau_w(d)$. Each type consistent (T, L) *represents* a type decorated tree-like structure $(T, \mathsf{bag}', \tau')$ of width at most m as follows. The domain of $\mathfrak{A}_{(T', \mathsf{bag}')}$ is the set of all equivalence classes $[w]_d$ with $w \in T$ and $d \in \mathsf{dom}_w$. The function τ' maps each domain element $[w]_d$ to $\tau_w(d)$, which is well-defined since (T, L) is type consistent. Finally, for every $w \in T$, the structure $\mathsf{bag}'(w) = (A(w), R_1^{\mathsf{bag}(w)}, \ldots, R_\ell^{\mathsf{bag}(w)})$ is defined by:

$$
\begin{aligned}
A(w) &= \{[w]_d \mid d \in \mathsf{dom}_w\}, \\
R_i^{\mathsf{bag}'(w)} &= \{([w]_{d_1}, \ldots, [w]_{d_j}) \mid (d_1, \ldots, d_j) \in R_i^{\mathsf{bag}_w}\} \quad \text{for } 1 \leq i \leq \ell.
\end{aligned}
$$

Conversely, for every type decorated tree-like structure (T, bag, τ) of width m, there is a Σ-labeled tree (T, L) that represents a type decorated tree-like structure $(T, \mathsf{bag}', \tau')$ such that there is an isomorphism π between $\mathfrak{A}_{(T,\mathsf{bag})}$ and $\mathfrak{A}_{(T,\mathsf{bag}')}$ that satisfies $\tau(d) = \tau'(\pi(d))$, for all $d \in \mathsf{dom}(T, \mathsf{bag})$. In fact, since Δ is of size $2m$, it is possible to select a mapping $\pi : \mathsf{dom}(T, \mathsf{bag}) \to \Delta$ such that for each $w \in T \setminus \{\varepsilon\}$ and each $d \in \mathsf{dom}(w) \setminus \mathsf{dom}(w \cdot -1)$, we have $\pi(d) \notin \{\pi(e) \mid e \in \mathsf{dom}(w \cdot -1)\}$. Define the Σ-labeled tree (T, L) by setting, for all $w \in T$, bag_w to the image of $\mathsf{bag}(w)$ under π and τ_w to the map defined by $\tau_w(h(d)) = \tau(d)$, for all $d \in \mathsf{dom}_w$. Clearly, π satisfies the desired properties.

The notion of a witness tree carries over straightforwardly from type decorated tree-like structures to type consistent Σ-labeled trees. In fact, one only needs to replace τ with τ_w in Condition $(*)$. Then, there is a witness tree for $\varphi(a)$ in a type consistent (T, L) iff there is a witness tree for $\varphi(a)$ in the type decorated tree-like structure $(T, \mathsf{bag}', \tau')$ represented by (T, L). The notion of properness also carries over straightforwardly. For easier reference, we spell it out explicitly below, and also replace the homomorphisms from the original formulation by witness trees as suggested by Lemma 12. A type consistent Σ-labeled tree (T, L) is *proper* if for all $w \in T$ and $a \in \mathsf{dom}_w$,

1′. for all $\exists x\, \varphi(x) \in \mathsf{cl}(\varphi_0)$, $\exists x\, \varphi(x) \in \tau_w(a)$ iff there is a $v \in T$, $b \in \mathsf{dom}_v$ with $\varphi(x) \in \tau_v(b)$;

2′. for all C2RPQs $\varphi(x) \in \mathsf{cl}(\varphi_0)$, $\varphi(x) \in \tau_w(a)$ iff there is a witness tree for $\varphi(a)$ in (T, L).

It is straightforward to verify that (T, L) is proper iff the type decorated tree-like structure $(T, \mathsf{bag}', \tau')$ represented by (T, L) is proper. Thus, our aim is to build a 2ATA $\mathcal{A}$ that accepts exactly the proper type consistent Σ-labeled trees (T, L) such that $\varphi_0 \in \tau_w(a)$ for some $w \in T$ and $a \in \mathsf{dom}_w$.

Automata construction. Let n be the size of φ_0, $k = n^2 + n$ the bound on the outdegree from Lemma 9, and assume from now on that the automata run over k-ary Σ-labeled trees. It is straightforward to construct a 2ATA $\mathcal{A}_0$ that accepts (T, L) iff it is type consistent and satisfies Condition 1′ of properness and the condition that $\varphi_0 \in \tau_w(a)$ for some $w \in T$ and $a \in \mathsf{dom}_w$. The number of states of the automaton is linear in the size of φ_0; details are omitted. We next show how to construct a 2ATA $\mathcal{A}_1 = (Q, \Sigma, q_0, \delta, F)$ that accepts a type consistent (T, L) iff Condition 2′ is satisfied. The automaton uses the set of states

$$Q = \{q_0\} \cup \{\varphi(\mathbf{a}), \overline{\varphi}(\mathbf{a}) \mid \varphi(\mathbf{a}) \in \mathsf{qcl}(\varphi_0, \Delta, n^2)\}.$$

where states of the form $\varphi(\mathbf{a})$ are used to verify the 'only if' part of Condition 2′ while states of the form $\overline{\varphi}(\mathbf{a})$ are used to verify the contrapositive of the 'if' part, that is, whenever a C2RPQ $\varphi(x) \in \mathsf{cl}(\varphi_0)$ is not in $\tau_w(a)$, then there is no witness tree for $\varphi(a)$ in (T, L).

Starting from the initial state, $\mathcal{A}_1$ loops over all nodes and domain elements using the following transitions, for all $(\mathsf{bag}, \tau) \in \Sigma$:

$$\delta(q_0, (\mathsf{bag}, \tau)) = \bigwedge_{1 \le i \le k} (i, q_0) \wedge \bigwedge_{a \in \mathsf{dom}(\mathsf{bag})} \Big(\bigwedge_{\substack{\varphi(x) \in \tau(a), \\ \varphi(x) \text{ a C2RPQ}}} \varphi(a) \wedge \bigwedge_{\substack{\neg\varphi(x) \in \tau(a), \\ \varphi(x) \text{ a C2RPQ}}} \overline{\varphi}(a) \Big)$$

We next give transitions for states of the form $\varphi(\mathbf{a}) \in \mathsf{qcl}(\varphi_0, \Delta, n^2)$. Informally, if the automaton visits a node w in state $\varphi(\mathbf{a})$, then this is an obligation to show that there is a witness tree whose root is labeled with $(w, \varphi(\mathbf{a}))$. In particular, the automaton has to demonstrate that there are suitable successors for the root of the witness tree, implementing Condition $(*)$. For a more concise definition of the transitions, we first establish a suitable notation. Let $\varphi(\mathbf{a}), \vartheta_1(\mathbf{a}_1), \ldots, \vartheta_\ell(\mathbf{a}_\ell) \in \mathsf{qcl}(\varphi_0, \Delta, n^2)$ and $(\mathsf{bag}, \tau) \in \Sigma$. We write $\varphi(\mathbf{a}) \to_{(\mathsf{bag},\tau)} \vartheta_1(\mathbf{a}_1), \ldots, \vartheta_\ell(\mathbf{a}_\ell)$ if there is a (Δ, n^2)-subdivision $\vartheta(\mathbf{a}')$ of $\varphi(\mathbf{a})$, a splitting

$\vartheta'_0(\mathbf{a}'_0), \dots, \vartheta'_\ell(\mathbf{a}'_\ell)$ of $\vartheta(\mathbf{a}')$, a homomorphism h from $\vartheta'_0(\mathbf{a}'_0)$ to bag given τ, and $\vartheta_i(\mathbf{a}_i)$ is obtained from $\vartheta'_i(\mathbf{a}'_i)$ by replacing each variable x in the domain of h with the constant $h(x)$; please note that this is an essential part of Condition $(*)$. Then, we include for each $\varphi(\mathbf{a}) \in \mathsf{qcl}(\varphi_0, \Delta, n^2)$ and each $(\mathsf{bag}, \tau) \in \Sigma$ the transition

$$\delta(\varphi(\mathbf{a}), (\mathsf{bag}, \tau)) = \bigvee_{\varphi(\mathbf{a}) \to_{(\mathsf{bag},\tau)} \vartheta_1(\mathbf{a}_1),\dots,\vartheta_\ell(\mathbf{a}_\ell)} \ \bigwedge_{1 \leq i \leq \ell} \ \bigvee_{j \in [k] \setminus \{0\}} (j, \vartheta_i(\mathbf{a}_i))$$

if $\mathbf{a} \subseteq \mathsf{dom}(\mathsf{bag})$ and set $\delta(\varphi(\mathbf{a}), (\mathsf{bag}, \tau)) = \mathsf{false}$ otherwise. States of the form $\overline{\varphi}(\mathbf{a})$ are treated dually, that is, using the transitions

$$\delta(\overline{\varphi}(\mathbf{a}), (\mathsf{bag}, \tau)) = \bigwedge_{\varphi(\mathbf{a}) \to_{(\mathsf{bag},\tau)} \vartheta_1(\mathbf{a}_1),\dots,\vartheta_\ell(\mathbf{a}_\ell)} \ \bigvee_{1 \leq i \leq \ell} \ \bigwedge_{j \in [k] \setminus \{0\}} (j, \overline{\vartheta}_i(\mathbf{a}_i))$$

if $\mathbf{a} \subseteq \mathsf{dom}(\mathsf{bag})$ and setting $\delta(\overline{\varphi}(\mathbf{a}), (\mathsf{bag}, \tau)) = \mathsf{true}$ otherwise.

To ensure that the witness trees constructed by the states of the form $\varphi(\mathbf{a})$ are finite, we use the parity condition $F = G_1, G_2$ with $G_1 = \mathsf{qcl}(\varphi_0, \Delta, n^2)$ and $G_2 = Q$. From an accepting run of $\mathcal{A}_1$ on an input tree (T, L), one can extract the witness trees that are required to show that the 'only if' direction of Condition 2′ is satisfied. Moreover, the run demonstrates that the witness trees forbidden by the 'if' direction do not exist. We thus obtain the following.

▶ **Lemma 13.** *The* $\text{UNFO}^{\mathsf{reg}}$ *sentence* φ_0 *is satisfiable iff* $L(\mathcal{A}_0) \cap L(\mathcal{A}_1)$ *is not empty.*

Putting together Lemmas 8, 11, and 13, it follows that satisfiability in $\text{UNFO}^{\mathsf{reg}}$ is in 2ExpTime. The corresponding lower bound is inherited from UNFO [46].

▶ **Theorem 14.** *In* $\text{UNFO}^{\mathsf{reg}}$*, satisfiability is* 2ExpTime*-complete.*

5 OMQ Evaluation and Containment

We study the complexity of OMQ evaluation and OMQ containment in $(\text{UNFO}^{\mathsf{reg}}, \text{UC2RPQ})$. Recall that the complexity of OMQ evaluation can be measured in different ways. In *combined complexity*, both the OMQ and the database on which it is evaluated are considered to be an input. In *data complexity*, the OMQ is fixed and the database is the only input. We first state our main result regarding the combined complexity of OMQ evaluation and the complexity of OMQ containment.

▶ **Theorem 15.** *In* $(\text{UNFO}^{\mathsf{reg}}, \text{UC2RPQ})$,
1. *OMQ evaluation is* 2ExpTime*-complete in combined complexity and*
2. *OMQ containment is* 2ExpTime*-complete.*

The upper bounds in Theorem 15 are a consequence of Lemmas 6 and 7 and Theorem 14. The lower bounds hold already when predicates are at most binary. For Point 1 this follows from the fact that OMQ evaluation is 2ExpTime-hard even for OMQs from the class $(\mathcal{ALCI}, \text{CQ})$ where the ontology is formulated in the description logic $\mathcal{ALCI}$, a fragment of UNFO with only unary and binary predicates, and the actual query is a CQ [39]. The same is true for Point 2 since in $(\mathcal{ALCI}, \text{CQ})$, OMQ evaluation can be reduced in polynomial time to OMQ containment in a straightforward way.

We next study the data complexity of $(\text{UNFO}^{\mathsf{reg}}, \text{UC2RPQ})$. A coNP lower bound is again inherited from (rather small) fragments of $(\text{UNFO}^{\mathsf{reg}}, \text{C2RPQ})$ [38, 23]. We give a coNP upper bound, thus establishing the following.

▶ **Theorem 16.** *OMQ evaluation in* (UNFO$^{\mathsf{reg}}$, UC2RPQ) *is* coNP*-complete in data complexity.*

Instead of directly considering OMQ evaluation, we work with a problem that we call database satisfiability. A database D is *satisfiable* with an UNFO$^{\mathsf{reg}}$ sentence φ if there is a model of φ that extends D. Let φ be an UNFO$^{\mathsf{reg}}$ sentence and Σ a set of predicate symbols. The *database satisfiability problem associated with* φ *and* Σ is to decide, given a Σ-database D, whether D is satisfiable with φ. Note that OMQ evaluation can be reduced in polynomial time to Boolean OMQ evaluation as in the proof of Lemma 6. Moreover, for a Boolean OMQ $Q = (\mathcal{O}, \Sigma, q)$ and a Σ-database D, $D \not\models Q$ iff D is satisfiable with $\mathcal{O} \wedge \neg q$. Consequently, a coNP upper bound for OMQ evaluation in (UNFO$^{\mathsf{reg}}$, UC2RPQ) can be proved by establishing an NP upper bound for database satisfiability in UNFO$^{\mathsf{reg}}$.

Let φ_0 be an UNFO$^{\mathsf{reg}}$ formula and Σ a set of predicate symbols. We may assume w.l.o.g. that φ_0 is normal and that every symbol from Σ occurs in φ_0. Subdivisions and splittings, defined as in Section 3, shall again play an inportant role. However, instead of subdividing an atom $\mathcal{A}(t, t')$ into at most n^2 many atoms, we use at most *two* intermediary points. Informally, this splits a witnessing path for $\mathcal{A}(t, t')$ into three parts: the first part is from t to the first element from D that appears on the path, the third subdivision atom represents the part from the last element from D that appears on the path to t', and the second atom represents the remaining middle part of the path.

We use $\mathsf{ecl}(\varphi_0)$ to denote the union of $\mathsf{cl}(\varphi_0)$ and $\mathsf{qcl}(\varphi_0)$, closed under single negation, where $\mathsf{qcl}(\varphi_0)$ is $\mathsf{qcl}(\varphi_0, \{x\}, 2)$ extended with the set of all $\mathcal{A}[q_0/s, F/\{s'\}](x, x)$ such that $\mathcal{A}$ is an NFA that occurs in φ_0 and s, s' are states in $\mathcal{A}$. An *extended* 1*-type for* φ_0 is a subset $t \subseteq \mathsf{ecl}(\varphi_0)$ such that t satisfies the conditions for being a 1-type from Section 3. We denote with $\mathsf{eTP}(\varphi_0)$ the set of all extended 1-types for φ_0.

Let D be a Σ-database. A *type decoration for* D is a mapping $\tau : \mathsf{dom}(D) \to \mathsf{eTP}(\varphi_0)$. We write $D, \tau \models \mathcal{A}(a, b)$ if $D \models \mathcal{A}(a, b)$ with the semantics of tests reinterpreted: instead of demanding $D \models \varphi(a')$ for a test $\varphi(x)?$ to hold at an element a', we now require that $\varphi(x) \in \tau(a')$. Let $\varphi(\mathbf{a})$ be an (instantiated) C2RPQ. A *homomorphism from* $\varphi(\mathbf{a})$ *to* D *given* τ is a function $h : \mathbf{a} \cup \mathsf{var}(\varphi) \to \mathsf{dom}(D)$ such that the following conditions are satisfied: $h(\mathbf{a}) = \mathbf{a}$, $h(\mathbf{t}) \in R^D$ for each $R(\mathbf{t}) \in \varphi(\mathbf{a})$, and for each $\mathcal{A}(t, t') \in \varphi(\mathbf{a})$, there are $a_1, \ldots, a_n \in \mathsf{dom}(D)$ and states $s_0, \ldots, s_n$ from $\mathcal{A}$, and a word $\nu_1 \cdots \nu_{n-1}$ from the alphabet of $\mathcal{A}$ such that

(a) $a_1 = h(t)$, $a_n = h(t')$, $s_0 = q_0$, and $s_n \in F$,

(b) $(a_i, a_{i+1}) \in R^D$ if $\nu_i = R$, $(a_{i+1}, a_i) \in R^D$ if $\nu_i = R^-$, and $\theta(x) \in \tau(a_i)$ and $a_{i+1} = a_i$ if $\nu_i = \theta(x)?$, for $1 \le i < n$, and

(c) $(s, \nu_i, s_{i+1}) \in \Delta$ for some s with $\mathcal{A}[q_0/s_i, F/\{s\}](x, x) \in \tau(a_{i+1})$, for $0 \le i < n$.

Note that Condition (c) admits the spontaneous change from state s_i to state s at a_{i+1}, without reading any of the ν_j symbols, when the atom $\mathcal{A}[q_0/s_i, F/\{s\}](x, x)$ is contained in $\tau(a_{i+1})$, asserting that we can indeed get from s_i to s starting at a_{i+1} and cycling back there while reading some unknown subword.

A type decoration τ is called *proper* if for all $a \in \mathsf{dom}(D)$, the following hold:

1. $\bigwedge_{\psi(x) \in \tau(a)} \psi(a)$ is satisfiable;

2. $\exists x \varphi(x) \in \tau(a)$ iff $\exists x \varphi(x) \in \tau(b)$, for all $a, b \in \mathsf{dom}(D)$ and all $\exists x \varphi(x) \in \mathsf{cl}(\varphi_0)$;

3. if $\neg\psi(x) \in \tau(a)$ for some $\psi(x) \in \mathsf{qcl}(\varphi_0)$, then for each $(\mathsf{dom}(D), 2)$-subdivision $\vartheta(\mathbf{a})$ of $\psi(a)$ and each splitting $\vartheta_0(\mathbf{a}_0), \vartheta_1(a_1), \ldots, \vartheta_\ell(a_\ell)$ of $\vartheta(\mathbf{a})$ such that there is a homomorphism h from $\vartheta_0(\mathbf{a}_0)$ to D given τ, there is an $i \in \{1, \ldots, k\}$ such that $\neg\vartheta_i(x) \in \tau(a_i)$.

Our NP procedure for database satisfiability is, given a Σ-database D, to guess a type decoration τ for D and to then verify in deterministic polynomial time that D is proper.

Note that the size of a type decoration is $\mathcal{O}(c \cdot |D|)$ for some constant c. The satisfiability checks in Point 1 of properness concern sentences whose size is independent of D, thus they need only constant time. Point 2 can be checked in time quadratic in the size of D. For Point 3, note that there are only polynomially many $(\mathsf{dom}(D), 2)$-subdivisions and splittings (in the size of D). To check the existence of the required homomorphism h, we can go through all candidates, directly verifying the homomorphism condition for relational atoms and proceedings as follows for NFA atoms: first extend D by exhaustively adding 'implied facts' of the form $\mathcal{A}(a, b)$, also taking into account assertions of the form $\mathcal{A}[q_0/s_i, F/\{s\}](x, x)$ that occur in τ-labels, as in Condition (c) above, and then treat NFA atoms like relational atoms. The following lemma finishes the proof of Theorem 16.

▶ **Lemma 17.** *D is satisfiable with φ_0 iff D has a proper type decoration τ such that $\varphi_0 \in \tau(a_0)$ for some $a_0 \in \mathsf{dom}(D)$.*

6 Model Checking

We show that model checking in UNFO$^{\mathsf{reg}}$ is complete for $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$, the class of problems that can be solved in polyniomial time given access to an NP oracle, but with only $O(\log^2 n)$ many oracle calls admitted. It thus has the same complexity as model checing in UNFO. Formally, the model checking problem for UNFO$^{\mathsf{reg}}$ is as follows: given a finite structure $\mathfrak{A}$ and a UNFO$^{\mathsf{reg}}$ sentence φ, does $\mathfrak{A} \models \varphi$ hold? Without tests in path expressions, UNFO$^{\mathsf{reg}}$ model checking can easily be reduced to model checking in UNFO: simply extend the input structure by exhaustively adding 'implied facts' of the form $\mathcal{A}(a, b)$ and then replace every $\mathcal{A}$ with a fresh binary relation symbol in both φ and $\mathfrak{A}$, obtaining an instance of UNFO model checking. With tests, this does not work. We would need multiple calls to UNFO model checking, essentially one call for every subformula inside a test in the input formula, but this bring us outside of $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$. We thus resort to expanding the $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$upper bound proof from [46], which is by reduction to a $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$-complete circuit value problem.

▶ **Theorem 18.** *The* UNFO$^{\mathsf{reg}}$ *model checking problem is* $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$*-complete.*

7 Conclusion

We have proved that OMQ evaluation in (UNFO$^{\mathsf{reg}}$, UC2RPQ) is decidable, 2ExpTime-complete in combined complexity, and coNP-complete in data complexity, and that OMQ containment and satisfiability are also 2ExpTime-complete. There are several interesting topics for future work. First, in contrast to UNFO, UNFO$^{\mathsf{reg}}$ does not have the finite model property and thus it would be interesting to study OMQ evaluation over finite models as well as finite satisfiability. Second, there are various natural directions for further increasing the expressive power. For example, one could allow any UNFO$^{\mathsf{reg}}$ formula with two free variables as a base case in regular path expressions instead of only atomic formulas. Such a logic would be strictly more expressive than propositional dynamic logic (PDL) with converse and intersection [31] and it would push the expressive power of UNFO$^{\mathsf{reg}}$ into the direction of regular queries, which have recently been proposed as an extension of C2RPQs [42]. Another natural extension was proposed by a reviewer of this paper: replace C2RPQs with linear Datalog to remove the asymmetry between binary relations and relations of higher arity in UNFO$^{\mathsf{reg}}$. Additional relevant extensions could arise from the aim to capture additional description logics. From this perspective, it would for example be natural to extend UNFO$^{\mathsf{reg}}$ with constants, with fixed points, and with so-called role inclusions, please see [5]. Since functional relations and similar forms of counting play an important role in description logics, we remark that it is implicit in [46] that satisfiability (and thus OMQ evaluation) is undecidable in UNFO extended with two functional relations. Finally, it would be interesting to inverstigate the complexity of OMQ containment in $(\text{UNFO}^{\mathsf{reg}}, \text{C2RPQ})$ without the restriction to a single ontology and to the full data signature. For (UNFO, CQ), a 2NExpTime upper bound can be proved by a slight adaptation of the technique in [21], also using (a slightly refined version of) the translation from (UNFO, CQ) to monadic disjunctive Datalog from [19]. However, accommodating C2RPQs in this approach seems nontrivial.

References

1. Antoine Amarilli, Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. Query answering with transitive and linear-ordered data. In *Proc. IJCAI*, pages 893–899, 2016.
2. Renzo Angles and Claudio Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008.
3. Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. IJCAI*, pages 446–451, 1991.
4. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press, 2nd edition, 2007.
5. Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambride University Press, 2017.
6. Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.
7. Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Proc. IJCAI*, pages 712–717, 2011.
8. Vince Bárány, Georg Gottlob, and Martin Otto. Querying the guarded fragment. *Logical Methods in Computer Science*, 10(2), 2014.
9. Vince Bárány, Balder ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, 2015.
10. Pablo Barceló. Querying graph databases. In *Proc. PODS*, pages 175–188, 2013.
11. Pablo Barceló, Gerald Berger, and Andreas Pieris. Containment for rule-based ontology-mediated queries. In *Proc. PODS*, 2018.
12. Meghyn Bienvenu, Diego Calvanese, Magdalena Ortiz, and Mantas Šimkus. Nested regular path queries in description logics. In *Proc. KR*, 2014.
13. Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in Horn description logics. In *Proc. IJCAI*, pages 965–971. IJCAI/AAAI Press, 2016.
14. Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Query containment in description logics reconsidered. In *Proc. KR*. AAAI Press, 2012.
15. Meghyn Bienvenu and Magdalena Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Proc. Reasoning Web*, volume 9203 of *LNCS*, pages 218–307. Springer, 2015.
16. Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Conjunctive regular path queries in lightweight description logics. In *Proc. IJCAI*, pages 761–767. IJCAI/AAAI, 2013.
17. Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Navigational queries based on frontier-guarded datalog: Preliminary results. In *Proc. AMW*, volume 1378 of *CEUR Workshop Proceedings*, 2015.

18 Meghyn Bienvenu, Magdalena Ortiz, and Mantas Šimkus. Regular path queries in lightweight description logics: Complexity and algorithms. *J. Artif. Intell. Res.*, 53:315–374, 2015.

19 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.

20 Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. How to best nest regular path queries. In *Proc. DL*, volume 1193 of *CEUR Workshop Proceedings*, pages 404–415, 2014.

21 Pierre Bourhis and Carsten Lutz. Containment in monadic disjunctive datalog, MMSNP, and expressive description logics. In *Proc. KR*, pages 207–216. AAAI Press, 2016.

22 Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR*, pages 70–80, 2008.

23 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *Artif. Intell.*, 195:335–360, 2013.

24 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In *Handbook of Automated Reasoning*, pages 1581–1634. Elsevier and MIT Press, 2001.

25 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proc. KR*, pages 176–185, 2000.

26 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. AAAI*, pages 391–396, 2007.

27 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Regular path queries in expressive description logics with nominals. In *Proc. IJCAI*, pages 714–720, 2009.

28 Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics via alternating tree-automata. *Inf. Comput.*, 237:12–55, 2014.

29 Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. A graphical query language supporting recursion. In *Proc. SIGMOD*, pages 323–330, 1987.

30 Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

31 Stefan Göller, Markus Lohrey, and Carsten Lutz. PDL with intersection and converse: satisfiability and infinite-state model checking. *J. Symb. Log.*, 74(1):279–314, 2009.

32 Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient algorithms for processing XPath queries. In *Proc. VLDB*, pages 95–106, 2002.

33 Georg Gottlob, Giorgio Orsi, Andreas Pieris, and Mantas Šimkus. Datalog and its extensions for semantic web databases. In *Proc. Reasoning Web*, volume 7487 of *LNCS*, pages 54–77. Springer, 2012.

34 Georg Gottlob, Andreas Pieris, and Lidia Tendera. Querying the guarded fragment with transitivity. In *Proc. ICALP II*, volume 7966 of *LNCS*, pages 287–298. Springer, 2013.

35 Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

36 Erich Grädel and Igor Walukiewicz. Guarded fixed point logic. In *Proc. LICS-99*, pages 45–54, 1999.

37 Roman Kontchakov and Michael Zakharyaschev. An introduction to description logics and query rewriting. In *Proc. Reasoning Web*, volume 8714 of *LNCS*, pages 195–244. Springer, 2014.

38 Adila Krisnadhi and Carsten Lutz. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. LPAR*, volume 4790 of *LNCS*, pages 333–347. Springer, 2007.

39 Carsten Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. IJCAR*, volume 5195 of *LNCS*, pages 179–193. Springer, 2008.

40 Magdalena Ortiz, Sebastian Rudolph, and Mantas Šimkus. Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In *Proc. IJCAI*, pages 1039–1044, 2011.

41 Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

42 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular queries on graph databases. *Theory Comput. Syst.*, 61(1):31–83, 2017.

43 Sebastian Rudolph and Markus Krötzsch. Flag & check: data access with monadically defined queries. In *Proc. PODS*, pages 151–162, 2013.

44 Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI*, pages 466–471, 1991.

45 Wiesław Szwast and Lidia Tendera. The guarded fragment with transitive guards. *Ann. Pure Appl. Logic*, 128(1-3):227–276, 2004.

46 Balder ten Cate and Luc Segoufin. Unary negation. *Logical Methods in Computer Science*, 9(3), 2013.

47 Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Proc. ICALP-98*, pages 628–641, 1998.