Containment of Graph Queries Modulo Schema

VÍCTOR GUTIÉRREZ-BASULTO, Cardiff University, UK
ALBERT GUTOWSKI, University of Warsaw, Poland
YAZMÍN IBÁÑEZ-GARCÍA, Cardiff University, UK
FILIP MURLAK, University of Warsaw, Poland

With multiple graph database systems on the market and a new Graph Query Language standard on the horizon, it is time to revisit some classic static analysis problems. Query containment, arguably the workhorse of static analysis, has already received a lot of attention in the context of graph databases, but not so in the presence of schemas. We aim to change this. Because there is no universal agreement yet on what graph schemas should be, we rely on an abstract formalism borrowed from the knowledge representation community: we assume that schemas are expressed in a description logic (DL). We identify a suitable DL that capture both basic constraints on the labels of incident nodes and edges, and more refined schema features such as participation, cardinality, and unary key constraints. Basing upon, and extending, the rich body of work on DLs, we solve the containment modulo schema problem for unions of conjunctive regular path queries (UCRPQs) and schemas whose descriptions do not mix inverses and counting. For two-way UCRPQs (UC2RPQs) we solve the problem under additional assumptions that tend to hold in practice: we restrict the use of concatenation in queries and participation constraints in schemas.

CCS Concepts: • **Theory of computation** → *Logic and databases.*

Additional Key Words and Phrases: conjunctive regular path queries, two-way, containment, schema, description logics, entailment, finite model reasoning

ACM Reference Format:
Víctor Gutiérrez-Basulto, Albert Gutowski, Yazmín Ibáñez-García, and Filip Murlak. 2024. Containment of Graph Queries Modulo Schema. *Proc. ACM Manag. Data* 2, 2 (PODS), Article 77 (May 2024), 26 pages. https://doi.org/10.1145/3651140

1 INTRODUCTION

Graph databases are today a mainstream technology with numerous applications in areas such as biology, social sciences, and logistics [35]. For example, in bioinformatics graph databases are commonly used to represent protein, cellular, and drug networks [25, 32]. Existing graph query languages, such as SPARQL or Cypher, and the upcoming Graph Query Language standard [19, 24] are navigational: they are build around regular path queries (RPQs) that allow one to test whether two nodes are related by a path of edges specified by a regular expression [1, 6, 13, 14]. Popular extensions of RPQs include: *two-way* RPQs (2RPQs), which can traverse edges forward and backwards; *conjunctive* RPQs (CRPQs), which are the closure of RPQs by conjunction and projection; and UC2RPQs, which combine both above extensions with closure under union. These formalisms have been widely studied in multiple classical contexts, one of which is static analysis. Arguably, the fundamental static analysis problem is *query containment*; that is, checking whether

Authors' addresses: Víctor Gutiérrez-Basulto, gutierrezbasultov@cardiff.ac.uk, Cardiff University, Cardiff, UK; Albert Gutowski, a.gutowski@mimuw.edu.pl, University of Warsaw, Warsaw, Poland; Yazmín Ibáñez-García, ibanezgarciay@cardiff.ac.uk, Cardiff University, Cardiff, UK; Filip Murlak, f.murlak@uw.edu.pl, University of Warsaw, Warsaw, Poland.

ACM 2836-6573/2024/5-ART77
https://doi.org/10.1145/3651140

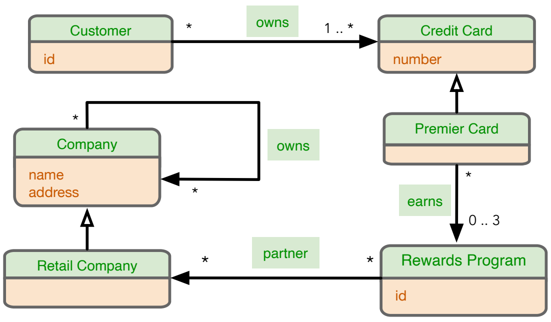

Fig. 1. Schema of a financial graph database

a query necessarily yields a subset of the result of another query. It was showed long ago that the containment problem for UC2RPQs is ExpSpace-complete [13], improving a previously established ExpSpace upper bound for CRPQs [23]. More recently, this result has been refined by investigating the containment problem of practical subclasses of CRPQs [22], or by identifying conditions under which the UC2RPQ-containment problem becomes tractable [21]. However, none of these works considers containment of navigational queries in the presence of schemas. A notable exception is the work by Deutsch and Tannen [20] in which expressive fragments of first-order logic are used as schema languages; the technical contribution of this work are decidability results, but no tight complexity bounds are provided. The only other exception we are aware of is a very recent work on graph database transformations [8], which shows as a side result that containment of UC2RPQs in *acyclic* UC2RPQs modulo schemas expressed in the Horn fragment of the description logic $\mathcal{ALCIF}$ is in 2EXPTIME.

The scarcity of work on graph query containment in the presence of schemas is perhaps explained by the fact that there is no agreement yet on what graph schemas should be. Recently, however, a community proposal called PG-Schema for the property graph data model has been put forward [2], leveraging an earlier constraint language called PG-Keys [3]. Motivated by this, we revisit the problem of containment of UC2RPQs modulo schema. Following the lead of Boneva et al. [8], we work with schemas expressed in description logics; specifically, in a logic called $\mathcal{ALCQI}$ which extends the basic Boolean-complete description logic $\mathcal{ALC}$ with the ability to count ($\mathcal{Q}$) and see edges backwards ($\mathcal{I}$) [5]. Description logics are the go-to formalism for conceptual modelling in the knowledge representation community, conveniently capturing ER models and UML class diagrams [4, 7, 15, 16]. $\mathcal{ALCQI}$ specifically has been advocated as a good choice [7, 15] and is well aligned with the core features of PG-Schema: over graphs with single labels on edges, $\mathcal{ALCQI}$ captures PG-Types (the core of PG-Schema) and a practically relevant subset of PG-Keys, including participation, cardinality, and unary key constraints (properties can be handled via reification). Among alternatives, $\mathcal{ALUNI}$ [16] cannot handle arbitrary combinations of node labels and DL-Lite [4] cannot talk about both endpoints of an edge simultaneously.

Example 1.1. Schema $\mathcal{S}$ in Figure 1 represents a conceptual data model where each customer owns at least one credit card; some credit cards are premier cards and earn rewards for purchases from partner retail companies and their subsidiaries; each premier card participates in at most 3 rewards programs. In Section 2 we show how to express this in $\mathcal{ALCQI}$. We retrieve customers and partners from which they earn rewards using $q_1(x,y) = (Owns \cdot Earns \cdot Partner \cdot Owns^*)(x,y)$ and $q_2(x,y) = (Owns{\cdot}Earns{\cdot}Partner)(x,z) \wedge RetailCompany(z) \wedge Owns^*(z,y)$; we use $\cdot$ for concatenation and * for unbounded iteration. Without schema, q_2 is contained in q_1, but not conversely; modulo $\mathcal{S}$, q_1 is contained in q_2 as well.

Using description logics as an abstract schema specification language allows us to reuse some ideas and techniques, but only up to a point. The main obstacle is the attitude to infinity: while database theory focuses on finite structures, knowledge representation traditionally embraces infinite models. For containment of navigational queries these two worlds are far apart: even for very limited logics, the answer depends on whether infinite graphs are allowed. Hence, previous results on query containment modulo constraints expressed in description logics [12, 17] do not apply in our case

Our contribution. We solve the containment modulo schema for the following combinations of query and schema languages:

(1) UCRPQs and either $\mathcal{ALCI}$ or $\mathcal{ALCQ}$,
(2) simple UC2RPQs and $\mathcal{ALCQ}$,
(3) UC2RPQs and $\mathcal{ALCQI}$ without participation constraints,

where 'simple' essentially means no concatenation in regular expressions. In all cases the problem is in 2EXPTIME; in items (1) and (2) the bound is tight by earlier results discussed below. We leave open the more general combination of UC2RPQs and $\mathcal{ALCQI}$: handling backward edges turned out to be rather subtle, both in queries and in schemas. However, the combinations we support already provide the expressivity needed in many settings. Item (2) is particularly useful. On one hand, recent studies of query logs [9, 10] show that a vast majority of queries are simple. On the other hand, the combination allows capturing some backward constraints by reversing the edges, because the query language supports backward edges; for example, one-to-many relationships can be supported.

Our approach, detailed in Section 3, relies on a non-trivial reduction of query containment to a related problem of *finite entailment*, asking if a given query is satisfied in every finite graph that extends a given graph and satisfies a given schema. Then, we extend significantly the limited available results on finite entailment [18, 27, 28] to be able to cover the combinations of schema and query languages we aim at (Sections 5–6). As a technical highlight, let us point out a general method of building structures that avoid a given UC2RPQ (Section 4), akin to the large-girth method of avoiding conjunctive queries [33], based on a novel construction of the *coil*.

2 PRELIMINARIES

Graphs. We fix a recursively enumerable set Γ of node labels, and an recursively enumerable set Σ of edge labels. We model graph databases as labeled directed graphs in which nodes can have multiple labels, while edges have a single label. Parallel edges are allowed, as long as they have different labels. We present such graphs as relational structures over unary relation symbols Γ and binary relation symbols Σ. That is, a graph G is a pair $\big(dom(G), \cdot^G\big)$ where $dom(G)$ 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 $r^G \subseteq dom(G) \times dom(G)$. 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 $r \in \Sigma$. Graph G is a *subgraph* of graph G', written $G \subseteq G'$, if $dom(G) \subseteq dom(G')$, $A^G \subseteq A^{G'}$ and $r^G \subseteq r^{G'}$ for all $A \in \Gamma$ and $r \in \Sigma$. A *homomorphism* h from graph G to graph G', written $h : G \to G'$, is a function $h : dom(G) \to dom(G')$ such that $u \in A^G$ iff $h(u) \in A^{G'}$ and $(u, v) \in r^G$ implies $\big(h(u), h(v)\big) \in r^G$ for all $u, v \in dom(G)$, $A \in \Gamma$, and $r \in \Sigma$; that is, h preserves the absence of node labels . We use $\bar{A}$ for *complement node labels*: we say a node has label $\bar{A}$ iff it does not have label A and let $\bar{A}^G = dom(G) \setminus A^G$, $\Gamma_0^- = \{\bar{A} \mid A \in \Gamma_0\}$, $\Gamma_0^\pm = \Gamma_0 \cup \Gamma_0^-$ for every $\Gamma_0 \subseteq \Gamma$. A *type* is a subset of $\Gamma^\pm$ that contains at most one of A and $\bar{A}$ for all $A \in \Gamma$. A *type over*

Γ_0 is a type that is a subset of $\Gamma_0^\pm$. A node u *is of type* τ *in* G if $u \in A^G$ for all $A \in \tau$. Thus, every node is of type $\emptyset$ and for every homomorphism $h : G \to G'$, if u is of type τ in G then $h(u)$ is of type τ in G'. A graph *realizes* type τ if it contains a node of type τ. A graph G *respects* a set Θ of types if each node in G is of some type from Θ. We use r^- for *inverse edges* and let $(r^-)^G = \{(u,v) \mid (v,u) \in r^G\}$, $\Sigma_0^- = \{r^- \mid r \in \Sigma_0\}$, $\Sigma_0^\pm = \Sigma_0 \cup \Sigma_0^-$ for every $\Sigma_0 \subseteq \Sigma$.

Queries. Let $\mathcal{V}$ be an enumerable set of variables. We work with *conjunctive two-way regular path queries* (C2RPQs) of the form

$$q = A_1(x_1) \wedge \cdots \wedge A_k(x_k) \wedge \varphi_1(y_1, z_1) \wedge \ldots \wedge \varphi_m(y_m, z_m),$$

where $x_1, \ldots, x_k, y_1, \ldots, y_m, z_1, \ldots, z_m \in \mathcal{V}$, $A_1, \ldots, A_k \in \Gamma^\pm$, and $\varphi_1, \ldots, \varphi_m$ are regular expressions over the alphabet $\Gamma^\pm \cup \Sigma^\pm$, using concatenation, union, and Kleene star; we allow complement node labels to facilitate query factorization in Section 3. We write $var(q)$ for the set of variables used in q. A *match* of q in a graph G is a function $\pi\colon var(q) \to dom(G)$ such that for every atom $A(x)$ in q, $\pi(x) \in A^G$, and for every atom $\varphi(y,z)$ in q there are $\ell_1, \ldots, \ell_n \in \Gamma^\pm \cup \Sigma^\pm$, and $v_0, \ldots, v_n \in dom(G)$ for some $n \in \mathbb{N}$ such that

(1) $v_0 = \pi(y)$ and $v_n = \pi(z)$;
(2) for all $i \in \{1, \ldots, n\}$, either $\ell_i \in \Sigma^\pm$ and $(v_{i-1}, v_i) \in (\ell_i)^G$ or $\ell_i \in \Gamma^\pm$ and $v_{i-1} = v_i \in (\ell_i)^G$;
(3) the word $\ell_1 \ldots \ell_n$ matches the regular expression φ.

We say that q is *satisfied* in G and write $G \models q$ if there is a match of q in G. Owing to homomorphisms preserving complement node labels, if $G \models q$ and G maps homomorphically to G' then $G' \models q$.

We also use *unions* of C2RPQs (abbreviated as UC2RPQs) represented as sets of C2RPQs $Q = \{q_1, \ldots, q_k\}$ and extend the notion of satisfaction to UC2RPQs in the natural fashion. By (unions of) conjunctive regular path queries, abbreviated as (U)CRPQs, we mean (U)2CRPQs that do not use labels from Σ^- in regular expressions. By (two-way) regular path queries, abbreviated as (2)RPQs, we mean binary atoms of C(2)RPQs. A query is *test-free* if it does not use labels from $\Gamma^\pm$ in regular expressions. A query is *simple* if it only uses regular expressions of the forms r and $(r_1 + r_2 + \cdots + r_n)^*$ with $r, r_1, \ldots, r_n \in \Sigma^\pm$.

Following [28], we sometimes work with UC2RPQs represented by means of a (nondeterministic) *semiautomaton* [26] $\mathcal{A} = (S, \Delta, \delta)$ where S is a finite set of states, $\Delta \subseteq \Gamma^\pm \cup \Sigma^\pm$ is a finite alphabet, and $\delta \subseteq S \times \Delta \times S$ is the transition relation. A semiautomaton is essentially a nondeterministic finite automaton without initial and final states; a run of a semiautomaton $\mathcal{A}$ over a word w is defined just like for a nondeterministic finite automaton, except that it can begin in any state and there is no notion of accepting runs. Under this representation, (2)RPQs are atoms of the form $\mathcal{A}_{s,s'}(t, t')$ where $s, s' \in S$ are states of $\mathcal{A}$. In the definition of a match we rephrase item (3) as follows:

(3') some run of $\mathcal{A}$ over $\ell_1 \ldots \ell_n$ begins in s and ends in s'.

Each UC(2)RPQ Q can be effectively rewritten as a UC(2)RPQ Q' expressed by means of a (nondeterministic) semiautomaton $\mathcal{A}$ of size linear in the total size of regular expressions in Q, by replacing each regular expression in Q with $\mathcal{A}_{s,s'}$ for some states s, s' of $\mathcal{A}$. Simple UC(2)RPQs correspond to disjoint unions of single-edge automata and single-state automata, with $\Delta \subseteq \Sigma^\pm$

Description logics. We work with graph properties expressed in the *description logic* $\mathcal{ALCQI}$ (and its fragments) [5]. In description logics, elements of Γ and Σ are called *concept names* and *role names*, respectively. $\mathcal{ALCQI}$ allows building more complex concepts with the following grammar:

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

where $A \in \Gamma^{\pm}$, $r \in \Sigma^{\pm}$, and $n \in \mathbb{N}$. We extend the interpretation function $\cdot^G$ to complex concepts as follows:

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

We also use additional operators that are redundant but useful when defining fragments; for brevity we introduce them as syntactic sugar: $\top := \neg\bot$, $C_1 \sqcup C_2 := \neg(\neg C_1 \sqcap \neg C_2)$, $\exists^{\geq n} r.C := \neg\exists^{\leq n-1} r.C$, $\exists r.C := \exists^{\geq 1} r.C$, $\forall r.C := \exists^{\leq 0} r.\neg C$. Statements in description logics have the form of *concept inclusions (CIs),*

$$C \sqsubseteq D$$

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 CIs is traditionally called a *TBox*. A graph G *satisfies* $\mathcal{T}$, written as $G \models \mathcal{T}$, if $G \models C \sqsubseteq D$ for each $C \sqsubseteq D \in \mathcal{T}$. A node v in G *satisfies* $C \sqsubseteq D$ if $v \in C^G$ implies $v \in D^G$; v *satisfies* $\mathcal{T}$ if it satisfies each $C \sqsubseteq D \in \mathcal{T}$.

In the logic $\mathcal{ALCQ}$ we disallow using inverse roles: in expressions of the form $\exists^{\leq n} r.C$ (and all derived expressions), we require that $r \in \Sigma$. In $\mathcal{ALCI}$, we disallow counting: in expressions $\exists^{\leq n} r.C$ we require $n = 0$; this corresponds to allowing only $\exists r.C$ and $\forall r.C$. If both above restrictions are imposed, we obtain the logic $\mathcal{ALC}$.

A TBox in each of these description logics can be *normalized*; that is, up to introducing auxiliary concept names, it can be expressed equivalently in the same logic using only CIs of the forms

$$K \sqsubseteq L\,, \quad A \sqsubseteq \exists r.B\,, \quad A \sqsubseteq \forall r.B\,, \quad A \sqsubseteq \exists^{\leq n} r.B\,, \quad A \sqsubseteq \exists^{\geq n} r.B\,,$$

where $A, B \in \Gamma^{\pm}$, $r \in \Sigma^{\pm}$, K is $\top$ or an intersection of concept names and complement concept names, and L is $\bot$ or a union of concept names and complement concept names (see e.g. [29, Prop. 1]).

Example 2.1. The following CIs capture key features of the schema $\mathcal{S}$ from Example 1.1 (with self-explanatory abbreviations):

$$\mathit{Customer} \sqsubseteq \exists \mathit{owns}.\mathit{CredCard}\,, \qquad \mathit{PremCC} \sqsubseteq \mathit{CredCard}\,,$$
$$\mathit{PremCC} \sqsubseteq \exists \mathit{earns}^{\leq 3}.\mathit{RwrdProg}\,, \qquad \mathit{RetlComp} \sqsubseteq \mathit{Comp}\,.$$

In order to fully capture schema $\mathcal{S}$ we also need to specify that only the depicted relationships are allowed. For instance,

$$\mathit{PremCC} \sqsubseteq \forall \mathit{earns}.\mathit{RwrdProg}\,, \qquad \mathit{RwrdProg} \sqsubseteq \forall \mathit{earns}^{-}.\mathit{PremCC}\,,$$

and similarly for *partner* and *owns*. We must also ensure that entities do not overlap, unless explicitly allowed by generalization relationships; for instance,

$$\mathit{Customer} \sqcap \mathit{CredCard} \sqsubseteq \bot\,, \qquad \mathit{RwrdProg} \sqcap \mathit{Company} \sqsubseteq \bot\,.$$

The use of inverse role earns^{-1} can be avoided, by flipping the CI to the contrapositive: $\overline{\mathit{PremCC}} \sqsubseteq \forall \mathit{earns}.\overline{\mathit{RwrdProg}}$.

3 CONTAINMENT AND ENTAILMENT

We focus on the Boolean variant of the containment problem. Given queries P and Q and a TBox $\mathcal{T}$, we write

$$P \subseteq_{\mathcal{T}} Q$$

if $G \models P$ implies $G \models Q$ for every finite graph G that satisfies $\mathcal{T}$. In the problem of *containment modulo schema* the input is P, Q, and $\mathcal{T}$, and the question to decide is whether $P \subseteq_{\mathcal{T}} Q$.

Without loss of generality we can focus on the containment of a connected C2RPQ in a union of connected C2RPQs; by a slight abuse of terminology, we call a UC2RPQ connected if it contains only connected C2RPQs. We also assume that the TBox is normalized, as explained in Section 2.

TBoxes without participation constraints

To warm up, let us see how to solve the containment problem for TBoxes that do not use *participation constraints*; that is, CIs of the form $A \sqsubseteq \exists r.B$ and $A \sqsubseteq \exists^{\geq n} r.B$. We rely on a simple model property established in [17] in the context of unrestricted satisfiability of C2RPQs in the presence of an $\mathcal{ALCIF}$ TBox. We use a reformulation of the simple model property given by Boneva et al. [8] in terms of Lee and Streinu's sparse graphs [31]. A finite connected graph G with n nodes and m edges is *c-sparse* for an integer $c \geq -1$ if $m \leq n + c$. A graph H *locally embeds* into graph G if there exists a homomorphism $h : H \longrightarrow G$ such that for all $r \in \Sigma^{\pm}$ and $(u, v_1), (u, v_2) \in r^H$, if $v_1 \neq v_2$ then $h(v_1) \neq h(v_2)$.

Theorem 3.1 (Boneva et al. [8]). *For every connected C2RPQ p and every graph G that satisfies p there exists a $|p|$-sparse graph G_p that satisfies p and locally embeds into G.*

Theorem 3.1 is the key to the containment problem. Indeed, for every UC2RPQ Q, if $G \not\models Q$, then $G_p \not\models Q$ because G_p maps homomorphically into G. Moreover, for every $\mathcal{ALCQI}$ TBox $\mathcal{T}$ without participation constraints, if $G \models \mathcal{T}$ then $G_p \models \mathcal{T}$. Indeed, let us fix a homomorphism witnessing that G_p locally embeds into G. With participation constraints forbidden, $\mathcal{T}$ can only contain CIs of three forms: $K \sqsubseteq L$, $A \sqsubseteq \forall r.B$, and $A \sqsubseteq \exists^{\leq n} r.B$. CIs of the form $K \sqsubseteq L$ carry over from G to G_p because homomorphisms preserve types of nodes. CIs of the form $A \sqsubseteq \forall r.B$ carry over because if a node in G_p has label A and an r-successor without label B, so does its image in G. CIs of the form $A \sqsubseteq \exists^{\leq n} r.B$ carry over because if a node in G_p has label A and more than n r-successors with label B, so does its image in G. Consequently, when deciding $p \subseteq_{\mathcal{T}} Q$ for $\mathcal{T}$ without participation constraints, it suffices to look for $|p|$-sparse counterexamples. This is easy because every $|p|$-sparse graph is a tree up to removing at most $|p| + 1$ edges and reversing edges so that they point away from the root. Assuming that the endpoints of each removed edge are indicated with unique markers, and reversed edges are suitably labelled, we can construct a tree automaton recognizing trees resulting from p-sparse counterexamples. To solve the containment problem, we test if the language recognized by the automaton is empty.

Theorem 3.2. *Containment of UC2RPQs modulo $\mathcal{ALCQI}$ TBoxes without participation constraints is in* 2EXPTIME.

The entailment problem

We approach containment modulo schemas with participation constraints through a related problem of entailment.

Given a finite graph G, a TBox $\mathcal{T}$, and a query Q, we say that Q is *entailed* by G and $\mathcal{T}$, written as $G, \mathcal{T} \models Q$, if $G' \models Q$ for every graph G' such that $G' \models \mathcal{T}$ and $G \subseteq G'$. We say that Q is *finitely entailed* by G and $\mathcal{T}$, written as $G, \mathcal{T} \models_{\mathit{fin}} Q$, if the above holds for every *finite* graph G'. The *finite entailment problem* is to decide if $G, \mathcal{T} \models_{\mathit{fin}} Q$ for given G, $\mathcal{T}$, and Q. (The traditional formulation uses a finite set of ground facts, called the ABox, instead of G.)

Finite entailment can be seen as a special case of containment modulo schema, via the well-known correspondence between conjunctive queries and graphs. As we will see, under certain assumptions on the TBox, also the converse reduction is possible. This will allow us to leverage existing knowledge on the finite entailment problem. The following results will be relevant.

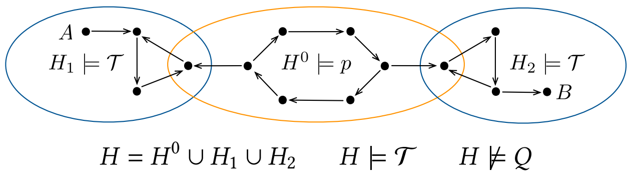
A
$H_1 \models \mathcal{T}$
$H^0 \models p$
$H_2 \models \mathcal{T}$
B
$H = H^0 \cup H_1 \cup H_2$
$H \models \mathcal{T}$
$H \not\models Q$

Fig. 2. A star-like countermodel in containment modulo schema (Lemma 3.5).

Theorem 3.3. *Finite entailment is* 2EXPTIME-*complete for the following combinations of query languages and description logics:*

(1) *test-free UCRPQs and $\mathcal{ALC}$ [28],*
(2) *UCQs with transitive atoms and $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ [27].*

The lower bound in (2) holds already for $\mathcal{ALC}$.

Recall that test-free means no labels from $\Gamma^{\pm}$ in regular expressions. UCQs with transitive atoms are simple UCRPQs that only use regular expressions of the form r and r^+ with $r \in \Sigma$.

Main results

Our main results are summarized in the following theorem.

Theorem 3.4. *Containment modulo schema is* 2EXPTIME-*complete for the following combinations of query and schema languages:*

(1) *UCRPQs and either $\mathcal{ALCI}$ or $\mathcal{ALCQ}$,*
(2) *simple UC2RPQs and $\mathcal{ALCQ}$.*

The 2EXPTIME lower bound of Theorem 3.3 carries over immediately to the containment of simple CRPQs modulo $\mathcal{ALC}$ TBox; the lower bounds in Theorem 3.4 follow. In order to establish the upper bounds, in the reminder of this section we show how to reduce containment modulo schema to finite entailment. Then, in Sections 4–6, we solve the finite entailment problem in the three cases necessary to establish Theorem 3.4.

From containment to entailment

We give an algorithm for containment modulo schema that uses finite entailment as an oracle. As for TBoxes without participation constraints, we first prove a simple countermodel property and then show how to decide if a simple countermodel exists.

We build upon Theorem 3.1. Essentially, we show that if we start from a graph G that satisfies $\mathcal{T}$, then G_p can be extended to a simple graph that satisfies $\mathcal{T}$ and still maps homomorphically to G, provided that $\mathcal{T}$ is an $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ TBox. A graph H is *star-like* if it consists of k disjoint graphs $H_1, \ldots, H_k$ (called *peripheral parts*) and a graph H^0 (called *the central part*) that shares exactly one node with each H_i for $i = 1, \ldots, k$ and the shared node has identical labels in both parts. In the lemma below, illustrated in Fig. 2, the central part corresponds to G_p and the peripheral parts are copies of G attached to provide witnesses for the participation constraints.

Lemma 3.5. *For every connected C2RPQ p, UC2RPQ Q, and $\mathcal{ALCI}$ or $\mathcal{ALCQ}$ TBox $\mathcal{T}$, $p \not\subseteq_{\mathcal{T}} Q$ iff there is a finite star-like graph H such that $H \models \mathcal{T}$, $H \not\models Q$, and*

- *the central part of H is $|p|$-sparse and satisfies p,*
- *all peripheral parts of H satisfy $\mathcal{T}$,*
- *in the central part, shared elements have only one incident edge and, in the $\mathcal{ALCQ}$ case, no outgoing edges.*

Proof. Let G be a counter-model. By Theorem 3.1 there is a $|p|$-sparse graph G_p that satisfies p and locally embeds into G via a homomorphism h. For each node u in G_p, each role $r \in \Sigma^{\pm}$ involved in a participation constraint in $\mathcal{T}$, and each r-successor v of $h(u)$ in G that is not an image via h of an r-successor of u in G_p, extend G_p by adding a fresh copy of G and an r-edge from u to the copy of v. Let H be the resulting graph with the copies of G treated as peripheral parts. Graph H still maps homomorphically to G, as h extends naturally. Hence, $H \not\models Q$. It is easy to verify that $H \models \mathcal{T}$: while one of the r^--successors of the copy of v is duplicated, we know that r is involved in a participation constraint. If $\mathcal{T}$ is an $\mathcal{ALCI}$ TBox, the duplication cannot be detected, because the logic does not count. If $\mathcal{T}$ is an $\mathcal{ALCQ}$ TBox, then $r^- \in \Sigma^-$, so the logic does not see r^--successors at all. The remaining three conditions from the statement of the lemma hold as well, by construction. □

To decide if such a graph H exists, we separate the existence of the central part from the existence of suitable peripheral parts. To this end, we replace global conditions $H \models \mathcal{T}$ and $H \not\models Q$ with local conditions over parts of H. We refer to this technique as *factorizing* the TBox $\mathcal{T}$ and the query Q.

To factorize Q, we replace it with a UC2RPQ $\widehat{Q}$ such that

(1) $\widehat{Q}$ is *factorized*; that is, $\widehat{Q}$ holds in a star-like graph iff it holds in any of its parts.

Of course, we cannot expect $\widehat{Q}$ to be equivalent to Q. We use fresh node labels in $\widehat{Q}$ and ensure that

(2) Q holds in a graph G iff $\widehat{Q}$ holds in every graph $\widehat{G}$ equal to G up to fresh node labels in $\widehat{Q}$.

This is sufficient to replace the condition $H \not\models Q$ in Lemma 3.5 with

- no part of H satisfies $\widehat{Q}$.

Indeed, by (2), $H \not\models Q$ iff $\widehat{H} \not\models \widehat{Q}$ for some $\widehat{H}$ obtained by labelling H with the fresh labels used in $\widehat{Q}$. By (1), $\widehat{H} \not\models \widehat{Q}$ iff no part of $\widehat{H}$ satisfies $\widehat{Q}$. As the additional labelling does not affect any other condition in Lemma 3.5, $\widehat{H}$ can play the role of H.

Example 3.6. Let Q consist of a single CRPQ

$$A(x) \wedge r^*(x,y) \wedge B(y)\,.$$

For $\widehat{Q}$ we can take the union of the following CRPQs:

$$A(x) \wedge \bar{C}_A(x)\,, \quad C_A(x) \wedge r^*(x,z) \wedge \bar{C}_A(z)\,, \quad C_A(z) \wedge C_B(z)\,,$$
$$\bar{C}_B(z) \wedge r^*(z,y) \wedge C_B(y), \quad \bar{C}_B(y) \wedge B(y)\,.$$

Intuitively, $\widehat{Q}$ detects if label C_A is missing in a node r-reachable from A or label C_B is missing in a node from which B is r-reachable, or some node has both label C_A and label C_B. Suppose that all edges in the graph H in Figure 2 have label r. Then, $H \not\models Q$. Let $\widehat{H}$ be obtained from H by adding label C_A to all nodes reachable from the unique node with label A, and label C_B to all nodes from which the unique node with label B can be reached. Then, $\widehat{H} \not\models \widehat{Q}$. Now, let $\widehat{H}$ be any graph obtained from H by adding labels C_A and C_B in some nodes. Suppose that $\widehat{H}$ satisfies $q_2 = C_A(x) \wedge r^*(x,z) \wedge \bar{C}_A(z)$. Then, on the path witnessing this there is an edge from a node with label C_A to a node without label C_A. It follows that some part of $\widehat{H}$ satisfies q_2. For other disjuncts of $\widehat{Q}$ the argument is either analogous or trivial. Hence, if $\widehat{H} \models \widehat{Q}$ then some part of $\widehat{H}$ satisfies $\widehat{Q}$. Note that complement node labels are crucial in this construction.

Lemma 3.7. *Given a connected UC2RPQ Q, a connected UC2RPQ $\widehat{Q}$ satisfying conditions (1) and (2) can be constructed in exponential time, while ensuring that $\widehat{Q}$ is a union of C2RPQs of polynomial size, and if Q is simple or one-way, so is $\widehat{Q}$.*

Proof. We begin by developing some auxiliary notions. A *pointed C2RPQ* (q, x) is a connected C2RPQ q along with a distinguished variable $x \in var(q)$. We say (q, x) *matches in graph* G *at node* v if there is a match π of q in G such that $\pi(x) = v$. Consider a match of a pointed C2RPQ (q, x) in a star-like graph G with central part G^0 and peripheral parts $G_1, G_2, \ldots, G_k$, with x matched in G^0. Intuitively, the match breaks (q, x) down into fragments matched in the respective parts of G. However, unlike for conjunctive queries, these fragments are not simply subsets of atoms of q, because paths witnessing 2RPQs can move back and forth between parts of G.

Let us split each 2RPQ $\mathcal{A}_{s,t}(y, z)$ in q into three 2RPQs,

$$\mathcal{A}_{s,s'}(y, y'), \ \mathcal{A}_{s',t'}(y', z'), \ \mathcal{A}_{t',t}(z', z)$$

for fresh variables y' and z', corresponding to three segments of the path witnessing $\mathcal{A}_{s,t}(y, z)$ in G: the maximal prefix within a peripheral part, the middle segment, and the maximal suffix within a peripheral part. (If either of the segments is empty, adjust the split accordingly.) For each shared node, replace all variables mapped to it with a fresh variable. For $i = 1, 2, \ldots, k$, consider the pointed C2RPQ (q_i, y_i) collecting all atoms witnessed entirely within G_i, with y_i being the variable mapped to the shared node in G_i, and the pointed C2RPQ (q^0, x') collecting all remaining atoms (if none are left, take $\mathcal{A}_{s,s}(x', x')$ for any s), with x' being either x itself or the variable that replaced it. Note that each (q_i, y_i) matches in G_i at the shared node, whereas (q^0, x') need not match in G^0 as some of its 2RPQs may be witnessed by paths detouring into peripheral parts.

A *unary factor* of (q, x) is any pointed C2RPQ that can be obtained as (q_i, y_i) for some match of (q, x) in a star-like graph, as well as any pointed query of the form $\big(\mathcal{A}_{s,s'}(y, y), y\big)$.

We aim to discover matches of (q, x) based on information about matches of its unary factors (p, y), encoded in fresh node labels $C_{p,y}$ dubbed *permissions*. Let (q', x') be obtained from (q^0, x') above by adding atoms $C_{q_i,y_i}(y_i)$ for $i = 1, 2, \ldots, k$ and extending the underlying semiautomaton with 'shortcut' transitions from state s to state s' over node label $C_{\mathcal{A}_{s,s'}(y,y),y}$, to account for detours. A *central factor* of (q, x) is any pointed C2RPQ that can be obtained as (q', x') for some match of (q, x) in a star-like graph.

A unary factor of a connected C2RPQ q is a unary factor of any (q, x) with $x \in var(q)$. Factors are connected C2RPQs, and a unary factor of a unary factor of q is a unary factor of q itself. In the case of simple C2RPQs, detours to peripheral parts are pointless so we keep the underlying semiautomaton unchanged, thus ensuring that factors are also simple.

We can now define $\widehat{Q}$ for a connected UC2RPQ Q as the union of queries

$$p' \wedge \bar{C}_{p,y}(y'),$$

where (p, y) is a unary factor of some $q \in Q$ and (p', y') is a central factor of (p, y), and queries

$$C_{q,x}(x),$$

where $q \in Q$ and $x \in var(q)$. Note that $\widehat{Q}$ is connected, and if Q is simple or one-way, so is $\widehat{Q}$. Given Q, one can compute $\widehat{Q}$ in exponential time, and each C2RPQ in $\widehat{Q}$ has polynomial size. It is routine, if a bit tedious, to check that $\widehat{Q}$ satisfies conditions (1) and (2). □

Factorizing the TBox is easy. Let $\mathcal{T}_0$ be $\mathcal{T}$ with all participation constraints dropped. Because we assume that each peripheral part of H satisfies $\mathcal{T}$, we can replace the condition $H \models \mathcal{T}$ with

- the central part of H satisfies $\mathcal{T}_0$ and each of its nodes satisfies all participation constraints in $\mathcal{T}$ unless it is shared with a peripheral part.

The only delicate point is that CIs of the form $A \sqsubseteq \exists^{\leq n} r.B$ with $r \in \Sigma$ carry over from the parts of H to the whole H. This is the case because, by the additional condition for $\mathcal{ALCQ}$ in the third item of Lemma 3.5, all outgoing edges of any node belong to a single part of H. Note that for $\mathcal{ALCQI}$ this would not work, as we would also have to deal with incoming edges.

Let $\mathsf{Tp}(\mathcal{T}, \widehat{Q})$ be the set of maximal types over labels used in $\mathcal{T}$ and $\widehat{Q}$, realized in finite graphs that satisfy $\mathcal{T}$ but not $\widehat{Q}$. We can now reformulate the criterion from Lemma 3.5 as follows: $p \not\subseteq_{\mathcal{T}} Q$ iff there is a $|p|$-sparse graph H_0 such that $H_0 \models p$, $H_0 \models \mathcal{T}_0$, $H_0 \not\models \widehat{Q}$, and each node violating a participation constraint from $\mathcal{T}$ is of some type from $\mathsf{Tp}(\mathcal{T}, \widehat{Q})$, and has only one incident edge and, in the $\mathcal{ALCQ}$ case, no outgoing edges. Given $\mathsf{Tp}(\mathcal{T}, \widehat{Q})$, we can adapt the automata-based argument behind Theorem 3.2 to decide if such H_0 exists. It remains to compute $\mathsf{Tp}(\mathcal{T}, \widehat{Q})$. For that we note that $\tau \in \mathsf{Tp}(\mathcal{T}, \widehat{Q})$ iff $G_\tau, \mathcal{T} \not\models_{fin} \widehat{Q}$, where G_τ is a graph consisting of a single isolated node of type τ. That is, we can compute $\mathsf{Tp}(\mathcal{T}, \widehat{Q})$ by solving an instance of finite entailment for each maximal type over labels used in $\mathcal{T}$ and $\widehat{Q}$. This completes the reduction of containment modulo schema to finite entailment. Note that we have actually showed that it suffices to solve a variant of finite entailment that asks if a given type $\tau \in \mathsf{Tp}(\mathcal{T}, \widehat{Q})$ can be realized in a finite graph G such that $G \models \mathcal{T}$ and $G \not\models Q$.

4 ASSEMBLING COUNTERMODELS

In Section 3, we solved containment modulo schema using a reduction to finite entailment, relying on a star-like countermodel property and on factorizing queries and TBoxes. Our approach to finite entailment is similar, but we reduce to simpler instances of the same problem, and countermodels have richer structure. In this section we prepare our tools. We begin with *concrete frames*, which will be used in Sections 5–6 to represent and manipulate complex decompositions of countermodels. Next, we lift our method of factorizing queries from star-like graphs to graphs represented by frames (TBoxes are handled in Sections 5–6). Finally, we pass to *abstract frames*, which will be used in Sections 5–6 to reduce a complex instance of finite entailment to multiple simpler ones.

In what follows we use the notion of a *pointed graph*, which is just a graph with a distinguished node. Two pointed graphs are considered isomorphic if they are isomorphic as graphs and the isomorphism preserves the distinguished node.

Concrete frames

A *concrete frame* is a finite graph without self-loops whose nodes represent disjoint components of a graph and edges represent edges between these components. More precisely, each frame node f is labelled with a pointed graph G_f with distinguished node v_f, and each edge originating in a frame node f is labelled with a pair (v, r) where $v \in dom(G_f)$ and $r \in \Sigma^{\pm}$. We assume that $dom(G_f) \cap dom(G_e) = \emptyset$ whenever $f \neq e$, and that different edges with labels (v, r) and (v, s) have different targets. For every frame node f and node $v \in dom(G_f)$ we define $G_{f,v}$ as the pointed graph obtained as follows. For the distinguished node we take v with labels inherited from G_f. For each edge from f to e with label (v, r) we add to $G_{f,v}$ the distinguished element v_e of G_e (with labels inherited from G_e) and add an r-edge from v to v_e. If $r = s^- \in \Sigma^-$, this results in an s-edge from v_e to v. We call graphs G_f the *components* and $G_{f,v}$ the *connectors* of the frame. While components are arbitrary pointed graphs, connectors are very simple pointed graphs with a single edge between the distinguished node and each non-distinguished node, no loops, and no edges between non-distinguished nodes. Every concrete frame F *represents* a graph G_F, obtained by taking the union of all its components and connectors. Edges in G_F that result from connectors are called *frame edges*; a frame edge and the corresponding edge in F may have opposite directions.

A concrete frame F *realizes* a type τ if the distinguished node of some component of F is of type τ. Clearly, if F realizes type τ, so does G_F.

Factorizing queries over concrete frames

We aim to replace the condition $G_F \not\models Q$ by a *local property* of frame F, depending exclusively on the set of isomorphism types of the components and connectors of F. In particular, we cannot depend on how the components and connectors are arranged in F.

Let Q be a connected UC2RPQ. A graph G *refutes* Q if $G \not\models \widehat{Q}$ (in particular, $G \not\models Q$). A concrete frame F *weakly refutes* Q if

- each component G_f refutes Q; that is, $G_f \not\models \widehat{Q}$;
- each connector $G_{f,v}$ refutes Q; that is, $G_{f,v} \not\models \widehat{Q}$.

We say that F *actually refutes* Q if G_F refutes Q. 'Weakly refuting Q' is a local property and it works for frames that are trees.

Lemma 4.1. *If a concrete frame is a tree and weakly refutes a connected UC2RPQ Q, then it actually refutes Q.*

Lemma 4.1 follows immediately from the lemma below applied to $\widehat{Q}$.

Lemma 4.2. *Let P be a factorized UC2RPQ and let F be a concrete frame that is a tree. Then $G_F \models P$ iff some component or connector of F satisfies P.*

Proof. We proceed by induction on the depth of the tree. Suppose $G_F \models P$. If F has depth 0, the unique component of F is equal to G_F, so it satisfies P. Otherwise, G_F is a star-like graph whose central part is the root component G_{f_0} of F and each peripheral part is a star-like graph whose central part is a connector $G_{f_0,v}$ for some v in G_{f_0}, and the peripheral parts are pointed graphs represented by subtrees of F rooted at children of f_0. Applying twice the fact that P is factorized, we conclude that P is satisfied either in G_{f_0}, or in $G_{f_0,v}$ for some $v \in G_{f_0}$ or in the graph represented by a subtree F' of F rooted at a child of f_0. In the first two cases we are done. In the third case, by the induction hypothesis, P is satisfied in a component or connector of F', and we are done too. □

In general, concrete frames that weakly refute Q need not actually refute Q, but some can be restructured to actually refute Q. To achieve this, we apply a novel general graph-theoretical construction which we describe next.

The coil

Our aim is to unravel a given graph sufficiently, without making it infinite. Indeed, the coil construction involves a bounded-recall (or sliding-window) unravelling of a graph. We think of paths as sequences of nodes and edges. Unless specified otherwise, paths are directed.

For a graph G and $n \geq 0$, let $\mathrm{Paths}(G, n)$ denote the set of paths (not necessarily simple) of length at most n in G, including paths of length 0 consisting of a single node. For a node v in G, let $\mathrm{Paths}(G, n, v)$ denote the set of paths in $\mathrm{Paths}(G, n)$ that originate in v.

For a node v in a graph G and $n > 0$, the graph $\mathrm{Unravel}(G, n, v)$ is a tree with nodes $\mathrm{Paths}(G, n, v)$ and an edge (π, π') whenever π' is an extension of π by one edge. The label of a node π in $\mathrm{Unravel}(G, n, v)$ is inherited from the last node of the path π, and the label of an edge (π, π') is inherited from the last edge of the path π'.

By the *n-suffix of a path π* we mean the suffix of length n of π if π has length at least n, and the whole path π otherwise. For a graph G and $n > 0$, $\mathrm{Coil}(G, n)$ is the graph with nodes $\mathrm{Paths}(G, n) \times \{0, \ldots, n\}$ and an edge $((\pi, \ell), (\pi', \ell'))$ whenever $\ell' \equiv \ell + 1 \pmod{(n+1)}$ and π' is the n-suffix of an extension of π by one edge. The label of a node (π, ℓ) in $\mathrm{Coil}(G, n)$ is inherited from the last node of π, and the label of an edge $((\pi, \ell), (\pi', \ell'))$ is inherited from the last edge of the path π'.

The coil has three key properties. Let $h_G : \mathrm{Coil}(G, n) \longrightarrow G$ map a node (π, ℓ) in $\mathrm{Coil}(G, n)$ to the last node on π.

Property 1. *The mapping $h_G : \mathrm{Coil}(G, n) \longrightarrow G$ is a surjective homomorphism.*

Proof. That h_G is a homomorphism follows directly from the construction of $\mathrm{Coil}(G, n)$. Surjectivity is witnessed by nodes $(\pi, 0)$ for paths π of length 0 consisting of just one node. □

Property 2. *For every node u in $\mathrm{Coil}(G, n)$, the subgraph induced by all nodes reachable from u by paths of length at most $n - 1$ is isomorphic to $\mathrm{Unravel}(G, n - 1, h_G(u))$.*

Proof. We begin with two preparatory observations. We can extend the function h_G to also map an edge $((\pi, \ell), (\pi', \ell'))$ in $\mathrm{Coil}(G, n)$ to the last edge on the path π'; note that it is an edge from $h_G((\pi, \ell))$ to $h_G((\pi', \ell'))$ in G. Then, for every node u in $\mathrm{Coil}(G, n)$, h_G induces a bijection between the edges outgoing from u and the edges outgoing from $h_G(u)$. This follows straight from the construction: for a node v in G, a path π ending in v, and an edge e outgoing from v, there is a unique n-suffix of the extension of π with the edge e.

We can also extend h_G to a length-preserving mapping from paths in $\mathrm{Coil}(G, n)$ to paths in G. By construction, for each edge $e = ((\pi, \ell), (\pi', \ell'))$ in $\mathrm{Coil}(G, n)$, we have $\ell' \equiv \ell + 1 \pmod{(n + 1)}$, and π' is the n-suffix of the extension of π with the edge $h_G(e)$. In consequence, for a path σ from (π, ℓ) to (π', ℓ') of length k in $\mathrm{Coil}(G, n)$, we have $\ell' \equiv \ell + k \pmod{(n + 1)}$, and π' is the n-suffix of the path obtained by concatenating π with $h_G(\sigma)$.

Now, consider a node $u' = (\pi', \ell')$ reachable from $u = (\pi, \ell)$ by a path σ of length at most n in $\mathrm{Coil}(G, n)$. Then, by the second observation above, the length of σ is $k = (\ell' - \ell) \bmod (n + 1)$ and $h_G(\sigma)$ is the k-suffix of π'. In the light of the first observation above, this means that σ is a unique path of length at most n from u to u'. It follows that the subgraph of $\mathrm{Coil}(G, n)$ induced by the set of nodes reachable from u by paths of length at most $n - 1$ is a tree. Using again the first observation above we show easily that this tree is isomorphic to $\mathrm{Unravel}(G, n - 1, h_G(u))$ via the mapping $u' \mapsto h_G(\sigma)$. □

For a node (π, ℓ) in $\mathrm{Coil}(G, n)$, we refer to the value ℓ as the *level* of the node. A subgraph of $\mathrm{Coil}(G, n)$ *visits level* ℓ if it contains a node of level ℓ.

Property 3. *Every connected subgraph H of $\mathrm{Coil}(G, n)$ that visits $k \leq n$ levels maps homomorphically to $\mathrm{Unravel}(G, k - 1, v)$ for some node v in G.*

Proof. Since H does not visit all levels and is connected, there exists a unique level ℓ_0 such that H visits level ℓ_0, but does not visit level $(\ell_0 - 1) \bmod (n + 1)$. For each node (π, ℓ) in H, let us call the value $(\ell - \ell_0) \bmod (n + 1)$ the *H-level of* (π, ℓ). The H-levels of nodes in H range from 0 to $k - 1$. A mapping g from (π, ℓ) to the suffix of π of length equal to the H-level of (π, ℓ) is the required homomorphism. For each edge $((\pi, \ell), (\pi', \ell'))$ in H, π' is the n-suffix of the extension of π with one edge; since the H-level of (π, ℓ) is not n, $g(\pi, \ell)$ is a prefix of $g(\pi', \ell')$; thus, since H is connected, there exists a unique node v in G such that $g(u)$ begins in v for all nodes u in H. It is straightforward to verify that g is indeed a homomorphism from H to $\mathrm{Unravel}(G, k - 1, v)$. □

Avoiding UC2RPQs

We aim to restructure a given concrete frame that weakly refutes Q into one that actually refutes Q, while preserving all local properties of the frame, such as weakly refuting a query. (We shall see later that all other properties required of countermodels amount to local properties of frames).

We call two concrete frames *locally isomorphic* if the sets of isomorphism types of their components and connectors are equal. Locally isomorphic concrete frames share local properties.

Lemma 4.3 below allows us to avoid UC2RPQs, just as the large-girth method does for conjunctive queries [33]. While in the case of conjunctive queries the construction depends only on the size of the queries, for UC2RPQs we rely on a more refined measure, defined below.

An undirected path in the graph G_F represented by a concrete frame F induces an undirected path in F. The *span of an undirected path* in frame F is the maximum absolute difference between the number of edges traversed forward and backward by an infix of the path. The *span of a 2RPQ* in F is the maximum span of undirected paths induced in F by paths witnessing the 2RPQ in G_F.

Lemma 4.3. *If a concrete frame F weakly refutes a connected UC2RPQ Q such that all 2RPQs in $\widehat{Q}$ have bounded span in F, then some frame locally isomorphic to F actually refutes Q.*

Proof. For a frame F and $n > 1$, let F_n be obtained from $\text{Coil}(F, n)$ by relabelling to ensure that all components have disjoint domains: for each node e in $\text{Coil}(F, n)$ with label G_e, change its label to a fresh isomorphic copy $\tilde{G}_e$ of G_e, and change labels in all outgoing edges from (v, r) to $(\tilde{v}, r)$, where $\tilde{v}$ is the copy of v in $\tilde{G}_e$. Properties 1 and 2 ensure that F_n is a frame and that it is locally isomorphic to F. The notion of levels of nodes carries over from $\text{Coil}(F, n)$ to F_n.

The homomorphism $h_F : \text{Coil}(F, n) \to F$ induces a homomorphism $\tilde{h}_F : G_{F_n} \to G_F$ that preserves frame edges; that is, an edge in G_{F_n} is a frame edge iff it is mapped by $\tilde{h}_F$ to a frame edge in G_F.

Claim 1. *The span of a 2RPQ in F_n is bounded from above by its span in F.*

Proof of Claim 1. Suppose that a witnessing path in G_{F_n} induces an undirected path σ in F_n. Because F_n differs from $\text{Coil}(F, n)$ only in the labelling of nodes and edges, we can view σ as an undirected path in $\text{Coil}(F, n)$; this change of perspective does not affect the span of σ. Applying $\tilde{h}_F$ to the witnessing path in G_{F_n}, we obtain a witnessing path in G_F such that the induced undirected path σ' in F can be obtained from σ by applying h_F. As the span of a path is preserved by homomorphisms, it follows that σ and σ' have the same span. This proves that the span of the 2RPQ in F bounds its span in F_n from above. □

Consider a match of a C2RPQ q in G_{F_n} along with witnessing paths for all 2RPQs. Let H be the subgraph of F_n built from all nodes f such that some witnessing path contains a node in G_f or some variable is mapped to a node in G_f, and all edges corresponding to the frame edges in G_{F_n} traversed by some witnessing path. We call H a *match of q in F_n*. Note that H is a frame and $G_H \models q$.

Claim 2. *If q is connected and consists of m 2RPQs, each of which has span in F_n bounded by k, then every match H of q in F_n visits at most $km + 1$ levels in F_n.*

Proof of Claim 2. The set of levels visited by an undirected path in F_n of span at most k is of the form $\{\ell + i \bmod n : i = 0, \ldots, d\}$ for some level ℓ and $d \leq k$. Because q is connected, H is a connected union of m undirected paths of span at most k. Consequently, the set of levels visited by H is of the form $\{\ell + i \bmod n : i = 0, \ldots, d\}$ for some level ℓ and $d \leq km$. □

We are now ready to prove the lemma. Let $m = \max\{|q| : q \in \widehat{Q}\}$ and let $k > 0$ be an upper bound on the span in F of all 2RPQs occurring in $\widehat{Q}$. We will show that F_{km+1} actually refutes Q.

Towards a contradiction, suppose that $G_{F_{km+1}} \models \widehat{Q}$. Then there is a match H of some q from $\widehat{Q}$ in F_{km+1}. Since Q is connected, so is $\widehat{Q}$ and H. By Claims 1-2, H visits at most $km + 1$ out of the $km + 2$ levels in F_{km+1}. Let H' be the subgraph of $\text{Coil}(F, km + 1)$ corresponding to H (equal to H up to relabelling). By Property 3, H' maps homomorphically to $\text{Unravel}(F, km, f)$ for some f in F. Let T be a frame obtained from $\text{Unravel}(F, km, f)$ by relabelling to ensure that all components have disjoint domains, as was done for F_{km+1}. It follows that G_H maps homomorphically to G_T, which means that $G_T \models \widehat{Q}$. But at the same time, T is locally isomorphic to a subframe of F, so T weakly refutes Q. Because T is a tree, by Lemma 4.1, T actually refutes Q. That is, $G_T \not\models \widehat{Q}$. The obtained contradiction shows that $G_{F_{km+1}} \not\models \widehat{Q}$. $\square$

Abstract frames

An *abstract frame F over $\Gamma_F \subseteq \Gamma$* is essentially a specification of a concrete frame: it is a finite graph without self-loops, just as a concrete frame, and its nodes and edges still represent components of a graph and edges between them, but the representation is symbolic rather than concrete.

Instead of a pointed graph G_f, each frame node f holds an abstract specification of a pointed graph, consisting of a set Θ_f of maximal types over Γ_F to be respected, a distinguished type $\tau_f \in \Theta_f$ to be realized, a TBox $\mathcal{T}_f$ to be satisfied, and a query Q_f to be avoided. We call $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$ an *(abstract) component* of F.

Each edge originating in frame node f is labelled with a pair (τ, r) for some $\tau \in \Theta_f$ and $r \in \Sigma^{\pm}$. Intuitively, it represents multiple edges originating in nodes of type τ. As in a concrete frame, different edges originating in f, labelled with (τ, r) and (τ, s) for any $r, s \in \Sigma^{\pm}$, must have different targets. Connectors $G_{f,\tau}$ are defined like for concrete frames, with v replaced by τ, except that we need to materialize the type τ and the types τ_e: we use fresh nodes u and u_e.

A *witnessing graph* for component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$ is any finite pointed graph G such that G respects Θ_f, the distinguished element of G is of type τ_f, $G \models \mathcal{T}_f$, and $G \not\models Q_f$. An abstract frame F *represents a concrete frame F'* if F' can be obtained from F by replacing each $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$ with an arbitrary witnessing graph G_f, and each edge labelled (τ, r) from f to e with edges labelled (v, r) from f to e, for v ranging over all nodes of type τ in G_f. (Note that connector $G_{f,v}$ is then isomorphic to $G_{f,\tau}$ for each node v of type τ in G_f.) An abstract frame F *represents a graph G* if it represents a concrete frame F' such that $G_{F'} = G$.

An abstract frame F realizes a type τ if $\tau \subseteq \tau_f$ for some component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$ of F. Clearly, if F realizes type τ, then so does every graph and every concrete frame represented by F.

An abstract component is *productive* if it has a witnessing graph. An abstract frame is *productive* if all its components are productive. Each productive abstract frame represents at least one concrete frame and at least one graph. Testing productivity of an abstract component is essentially a special case of finite entailment; testing if an abstract frame is productive leads to multiple instances of finite entailment.

To lift the notion of weakly refuting from concrete to abstract frames we rephrase the first condition as

- for each component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, if a graph does not satisfy Q_f, then it refutes Q; that is, Q_f contains $\widehat{Q}$ in the query containment sense;

and additionally require that Γ_F includes all node labels used in $\widehat{Q}$. It follows that if an abstract frame weakly refutes Q, so do all concrete frames it represents.

5 ENTAILMENT OF ONE-WAY QUERIES

In this section we establish the following result.

Theorem 5.1. *Finite entailment of UCRPQs in $\mathcal{ALCI}$ and $\mathcal{ALCQ}$ is* 2EXPTIME-*complete.*

The lower bound is inherited from finite entailment of simple UCRPQs in $\mathcal{ALC}$ [27]. The upper bound for $\mathcal{ALCQ}$ follows by eliminating tests from the query by encoding the type of each node in the label of each outgoing edge and careful inspection of the proof for $\mathcal{ALC}$ [28]. Here, we show the upper bound for $\mathcal{ALCI}$. As noted in Section 3, for Theorem 3.4 it suffices to decide if a given type τ (over labels used in $\mathcal{T}$ and $\widehat{Q}$) can be realized in a finite graph that satisfies $\mathcal{T}$ and refutes Q. As finite entailment is a special case of containment modulo schema, this also suffices for Theorem 5.1.

The general idea is to decompose countermodels into components in which it is enough to reason exclusively about forward (outgoing) edges or exclusively about backward (incoming) edges; this will ultimately allow us to reduce finite entailment in $\mathcal{ALCI}$ to finite entailment in $\mathcal{ALC}$. In order to confine RPQs to a limited number of components, while at the same time providing both forward and backward witnesses required in the $\mathcal{ALCI}$ TBox, we shall alternate between forward and backward components.

Alternating frames

To distinguish forward and backward components we use a fresh node label $C_{\rightarrow}$. For the sake of symmetry, we refer to $\bar{C}_{\rightarrow}$ as $C_{\leftarrow}$. In any graph, nodes with label $C_{\rightarrow}$ are called *forward* and those with label $C_{\leftarrow}$ are called *backward*. A concrete frame is *alternating* if

- every component G_f satisfies $\top \sqsubseteq C_{\rightarrow}$ or $\top \sqsubseteq C_{\leftarrow}$;
- every connector $G_{f,v}$ is *directed*, that is, all its edges are directed from backward nodes to forward nodes and either $C_{\rightarrow}$ or $C_{\leftarrow}$ occurs only in the distinguished node.

For abstract frames we replace the first condition with

- for every component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, either each type in Θ_f contains $C_{\rightarrow}$ or each type in Θ_f contains $C_{\leftarrow}$.

In a graph represented by an alternating frame, components have only incoming or only outgoing frame edges. Hence, an RPQ can only traverse a single frame edge; that is, its span is at most 1.

Factorizing $\mathcal{ALCI}$ TBoxes over alternating frames

In order to separate reasoning about forward and backward edges, we shall require that all forward witnesses of forward nodes be provided in components and all backward witnesses in connectors, and symmetrically for backward nodes. This leads to the following definition.

Let $\mathcal{T}_{\rightarrow}$ be an $\mathcal{ALC}$ TBox obtained from $\mathcal{T}$ by dropping all participation constrains involving inverse roles, that is, CIs of the form $A \sqsubseteq \exists r^{-}.B$, and flipping CIs involving universal restrictions over inverse roles: $A \sqsubseteq \forall r^{-}.B$ is replaced by $\bar{B} \sqsubseteq \forall r.\bar{A}$. The TBox $\mathcal{T}_{\leftarrow}$ is defined symmetrically. Note that while $\mathcal{T}_{\leftarrow}$ is not an $\mathcal{ALC}$ TBox, it uses only inverse roles. Hence, by treating inverse roles as role names, we can turn it into an $\mathcal{ALC}$ TBox. An alternating concrete frame *satisfies* $\mathcal{T}$ if

- each component G_f satisfies $\{\top \sqsubseteq C_{\rightarrow}\} \cup \mathcal{T}_{\rightarrow}$ or $\{\top \sqsubseteq C_{\leftarrow}\} \cup \mathcal{T}_{\leftarrow}$;
- in each connector $G_{f,v}$, the distinguished node satisfies $\{\top \sqsubseteq C_{\rightarrow}\} \cup \mathcal{T}_{\leftarrow}$ or $\{\top \sqsubseteq C_{\leftarrow}\} \cup \mathcal{T}_{\rightarrow}$.

For an alternating abstract frame F, we replace the first item with

- for each component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, either $\mathcal{T}_f$ entails $\mathcal{T}_{\rightarrow}$ and each type in Θ_f contains $C_{\rightarrow}$, or $\mathcal{T}_f$ entails $\mathcal{T}_{\leftarrow}$ and each type in Θ_f contains $C_{\leftarrow}$,

and additionally require that Γ_F includes all concept names used in $\mathcal{T}$. As connectors in alternating frames are directed and CIs are suitably flipped in $\mathcal{T}_{\leftarrow}$ and $\mathcal{T}_{\rightarrow}$, the second item implies that each connector satisfies all CIs of the form $A \sqsubseteq \forall r.B$ or $A \sqsubseteq \forall r^{-}.B$ in $\mathcal{T}$.

In the definition above we provide all backward witnesses to forward nodes and forward witnesses to backward nodes in connectors, even if some are also provided in components. This is correct because $\mathcal{ALCI}$ cannot detect duplicate witnesses.

Lemma 5.2. *If an alternating frame F satisfies an $\mathcal{ALCI}$ TBox $\mathcal{T}$, so does each graph F represents.*

Finite entailment by way of frames

We are now ready to characterize finite entailment in terms of abstract frames.

Lemma 5.3. *For a unary type τ, an $\mathcal{ALCI}$ TBox $\mathcal{T}$, and a connected UCRPQ Q, type τ is realized in a finite graph that satisfies $\mathcal{T}$ and refutes Q iff τ is realized in a productive alternating abstract frame that satisfies $\mathcal{T}$ and weakly refutes Q. [Proof: A.1]*

Using Lemma 5.3, we can compute the set of such types τ using a greatest fixed-point procedure (a variant of type elimination [30, 34]). Testing productivity of the abstract components of the witnessing frames is reduced to finite entailment of test-free CRPQs in $\mathcal{ALC}$ [28] (see Appendix A.2).

6 ENTAILMENT OF TWO-WAY QUERIES

We now move to two-way queries and show the following theorem.

Theorem 6.1. *Finite entailment of simple UC2RPQs in $\mathcal{ALCQ}$ is* 2EXPTIME*-complete.*

The lower bound follows from Theorem 3.3. Here we establish the upper bound. As in Section 5, we work with the variant of finite entailment that asks if a type τ can be realized in a finite graph that satisfies a TBox $\mathcal{T}$ and refutes a simple connected UC2RPQ Q.

The overall strategy is to successively reduce the number of role names used in the TBox until none are left. The case with no roles is easy (see Appendix B.1). In order to reduce the number of roles by one, we perform an intermediate step that neutralizes certain atoms in the query. For $\Sigma_0 \subseteq \Sigma$, by a *$\Sigma_0$-reachability atom* we mean an atom $(r_1 + r_2 + \cdots + r_k)^*(x, y)$ such that $\{r_1, r_2, \ldots, r_k\} \supseteq \Sigma_0$ or $\{r_1, r_2, \ldots, r_k\} \supseteq \Sigma_0^-$. Let $\Sigma_{\mathcal{T}}$ be the set of role names used in the input TBox $\mathcal{T}$. Intuitively, we want to reduce an instance of the problem to multiple instances that do not involve $\Sigma_{\mathcal{T}}$-reachability atoms, and then reduce each of those to multiple instances with fewer role names in the TBox. However, rather than modifying the query, we modify the problem. We say that a graph G *refutes Q modulo Σ_0-reachability* if $G \not\models \widehat{Q} \bmod \Sigma_0$ where $\widehat{Q} \bmod \Sigma_0$ is the query obtained from $\widehat{Q}$ by dropping all Σ_0-reachability atoms. The problem we will be solving, dubbed *finite entailment modulo Σ_0-reachability*, is to decide if a type τ can be realized in a finite graph that satisfies $\mathcal{T}$, respects Θ, and refutes Q modulo Σ_0-reachability, where $\Sigma_0 \supseteq \Sigma_{\mathcal{T}}$. (We recover the original problem by taking $\Theta = \{\emptyset\}$ and $\Sigma_0 = \Sigma_{\mathcal{T}} \cup \{r\}$ for some fresh role name r.) The intermediate step will amount to replacing Σ_0-reachability with $\Sigma_{\mathcal{T}}$-reachability. Next, we explain how to factorize $\mathcal{ALCQ}$ TBoxes over frames, which will be needed in both steps.

Factorizing $\mathcal{ALCQ}$ TBoxes

Let $\mathcal{T}$ be an $\mathcal{ALCQ}$ TBox. Let $\Gamma_{\mathcal{T}}$ be a set of fresh concept names $C_{n,r,D}$ for each role name r and concept name D involved in an at-least or at-most restriction in $\mathcal{T}$, and each $n \leq N$ where N is one plus the maximal number used in $\mathcal{T}$. Let $\mathcal{T}_=$ be the TBox obtained from $\mathcal{T}$ by dropping all CIs involving roles, and adding

$$\begin{aligned} C_{n,r,D} &\sqsubseteq \exists^{\geq n} r.D && \text{for each } C_{n,r,D} \in \Gamma_{\mathcal{T}}, \\ C_{n,r,D} &\sqsubseteq \exists^{\leq n} r.D && \text{for each } C_{n,r,D} \in \Gamma_{\mathcal{T}} \text{ with } n < N, \\ \bar{C}_{N,r,D} &\sqsubseteq \exists^{\leq n-1} r.D && \text{for each } C_{N,r,D} \in \Gamma_{\mathcal{T}}. \end{aligned}$$

For every graph that satisfies all CIs from $\mathcal{T}$ that do not involve an at-least restriction, there is a unique way to place labels $C_{n,r,D}$ that makes $\mathcal{T}_=$ satisfied. In particular, there is one for every subgraph of a graph that satisfies $\mathcal{T}$. We also define $\mathcal{T}_+$ as the TBox obtained from $\mathcal{T}$ by replacing each $C \sqsubseteq \exists^{\leq n} r.D$ with

$$C \sqsubseteq \bigsqcup_{i=0}^{n} \left(C_{i,r,D} \sqcap \exists^{\leq n-i} r.D\right)$$

and each $C \sqsubseteq \exists^{\geq n} r.D$ with

$$C \sqsubseteq \bigsqcup_{i=0}^{n} \left(C_{i,r,D} \sqcap \exists^{\geq n-i} r.D\right) \sqcup \bigsqcup_{i=n+1}^{N} C_{i,r,D} \,.$$

$\mathcal{T}_+$ can be normalized at the cost of introducing additional concept names, polynomially many in the size of $\mathcal{T}$ and N. A concrete frame *satisfies* an $\mathcal{ALCQ}$ TBox $\mathcal{T}$ if

- each component G_f satisfies $\mathcal{T}_=$;
- in each connector $G_{f,v}$, the distinguished node satisfies $\mathcal{T}_+$ and has no incoming edges.

For abstract frames we rephrase the first condition as

- for each component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, $\mathcal{T}_f$ entails $\mathcal{T}_=$.

LEMMA 6.2. *If a frame satisfies an $\mathcal{ALCQ}$ TBox $\mathcal{T}$, each graph it represents satisfies $\mathcal{T}$.*

From Σ_0-reachability to $\Sigma_{\mathcal{T}}$-reachability

Without loss of generality we can assume that countermodels only use role names from $\Sigma_{\mathcal{T}}$. The general idea of this step is to decompose countermodels into strongly connected components. Within each component, all $\Sigma_{\mathcal{T}}$-reachability atoms are trivially satisfied, so if the component refutes Q modulo Σ_0-reachability, it also refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability (because $\Sigma_{\mathcal{T}} \subseteq \Sigma_0$). Replacing Σ_0-reachability with $\Sigma_{\mathcal{T}}$-reachability makes the conditions imposed on witnessing countermodels more demanding, but we are guaranteed to find such countermodels if any countermodels exist. A delicate aspect here is that we will be testing the existence of suitable components by invoking a simpler instance of the decision problem, and there is no way to ensure that they are strongly connected. The point is, however, that this is not needed: any component, strongly connected or not, that refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability, also refutes Q modulo Σ_0-reachability (provided $\Sigma_{\mathcal{T}} \subseteq \Sigma_0$). This leads to the following strengthening of the notion of weakly refuting.

For a connected simple UC2RPQ Q and $\Sigma_1 \subseteq \Sigma_2 \subseteq \Sigma$, a concrete frame *weakly refutes Q modulo (Σ_1, Σ_2)-reachability* if

- each component G_f refutes Q modulo Σ_1-reachability; that is, $G_f \not\models \widehat{Q} \bmod \Sigma_1$;
- each connector $G_{f,v}$ refutes Q modulo Σ_2-reachability; that is, $G_{f,v} \not\models \widehat{Q} \bmod \Sigma_2$.

For an abstract frame F we rephrase the first condition as

- for each component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, Q_f contains $\widehat{Q} \bmod \Sigma_1$ in the query containment sense;

and require that Γ_F includes all node labels used in $\widehat{Q}$.

An abstract frame *respects* a set Θ of types, if for each component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, every type from Θ_f contains a type from Θ.

LEMMA 6.3. *Consider a type τ, an $\mathcal{ALCQ}$ TBox $\mathcal{T}$, a set Θ of types, a connected simple UC2RPQ Q, and $\Sigma_0 \supseteq \Sigma_{\mathcal{T}}$. Type τ is realized in a finite graph that satisfies $\mathcal{T}$, respects Θ, and refutes Q modulo Σ_0-reachability iff τ is realized in a productive abstract frame that is a tree, satisfies $\mathcal{T}$, respects Θ, and weakly refutes Q modulo $(\Sigma_{\mathcal{T}}, \Sigma_0)$-reachability. [Proof: B.2]*

Witnessing abstract frames from Lemma 6.3 can be constructed bottom-up from suitable productive abstract components (Appendix B.3). Testing productivity of an abstract component amounts to finite entailment modulo $\Sigma_{\mathcal{T}}$-reachability, which we handle next.

Entailment modulo $\Sigma_{\mathcal{T}}$-reachability

This time we decompose countermodels into components that use one role name fewer. In different components we eliminate different role names, in a round-robin fashion, ensuring that the span of every simple 2RPQ is at most $|\Sigma_{\mathcal{T}}|$, unless it is a $\Sigma_{\mathcal{T}}$-reachability atom.

Let $r_1, r_2, \ldots, r_m$ be an enumeration of $\Sigma_{\mathcal{T}}$. For convenience, let $r_{m+1} = r_1$. For each $r \in \Sigma_{\mathcal{T}}$, let C_r be a fresh concept name. By an r-node we mean a node that has label C_r and labels $\bar{C}_s$ for all $s \in \Sigma_{\mathcal{T}} \setminus \{r\}$. A concrete frame is *role-alternating* if

- every component G_f satisfies
$$\{\top \sqsubseteq C_r\} \ \cup \ \left\{\top \sqsubseteq C_{0,r,D} \,\middle|\, C_{0,r,D} \in \Gamma_{\mathcal{T}}\right\} \ \cup \ \left\{\top \sqsubseteq \bar{C}_s \,\middle|\, s \in \Sigma_{\mathcal{T}} \setminus \{r\}\right\}$$
for some $r \in \Sigma_{\mathcal{T}}$;
- every connector $G_{f,v}$ is *role-directed*, that is, for some i, the distinguished node is an r_i-node, all remaining nodes are r_{i+1}-nodes, and all edges are r_i-edges originating in the distinguished node.

For abstract frames we replace the first condition with

- for every component $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$, there is $r \in \Sigma_{\mathcal{T}}$ such that each $\tau \in \Theta_f$ contains C_r, all $C_{0,r,D} \in \Gamma_{\mathcal{T}}$, and all $\bar{C}_s$ with $s \in \Sigma_{\mathcal{T}} \setminus \{r\}$.

Being role-alternating is a local property of a frame: it is inherited by locally isomorphic frames.

Lemma 6.4. *For every role-alternating concrete frame F, the span in F of a simple 2RPQ that is not a $\Sigma_{\mathcal{T}}$-reachability atom, is at most $|\Sigma_{\mathcal{T}}|$. [Proof: B.4]*

Lemma 6.5. *Consider a type τ, an $\mathcal{ALCQ}$ TBox $\mathcal{T}$, a set Θ of types, and a connected simple UC2RPQ Q. Type τ is realized in a finite graph that satisfies $\mathcal{T}$, respects Θ, and refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability iff τ is realized in a productive role-alternating abstract frame that satisfies $\mathcal{T}$, respects Θ, and weakly refutes Q modulo $(\Sigma_{\mathcal{T}}, \Sigma_{\mathcal{T}})$-reachability. [Proof: B.5]*

The existence of witnessing abstract frame from Lemma 6.5 can be decided by a greatest fixed-point procedure, as in Section 5 (see Appendix B.6). Testing productivity of the involved abstract components reduces to an instance of finite entailment modulo $\Sigma_{\mathcal{T}}$. Each component of a role-alternating abstract frame that satisfies $\mathcal{T}$, effectively forbids some role r indicated by the concept name C_r present in the type of the distinguished node. We can eliminate role r entirely from the TBox, arriving at an instance of finite entailment modulo $\Sigma_{\mathcal{T}}$ with one role fewer. Apart from that, the reduction only affects the type τ to realize and the set Θ of allowed types, which must account for newly added concept names from $\Gamma_{\mathcal{T}}$. The size of $\Gamma_{\mathcal{T}}$ is exponential in the size of the original TBox (not the current TBox $\mathcal{T}$ that is the result of previous reductions). This allows us to use a recursive call to our decision procedure, without additional blowup (see Appendix B.7).

7 DISCUSSION

In this paper we have made significant progress on the graph query containment problem modulo schemas. Along the way, we also extended existing results on finite entailment of graph queries in expressive description logics. We believe that our methods can be adapted to handle reasoning about parallel edges with different labels and to multiple labels over edges, possibly at the cost of increased complexity. Handling full $\mathcal{ALCQI}$ schemas, on the other hand, is much more challenging. Similarly, handling full UC2RPQs, even for the basic logic $\mathcal{ALC}$ seems to require new ideas. Besides this, it would also be interesting to develop techniques that are better suited for implementation, since our current approach relies on automata- and type-based techniques which are always worst-case complexity. Another direction is ontology-mediated query containment [11] of navigational queries. In this setting, it would be already challenging to consider queries that only allow reachability.

ACKNOWLEDGMENTS

Albert Gutowski and Filip Murlak were supported by Poland's NCN grant 2018/30/E/ST6/00042.

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* 50, 5 (2017), 68:1–68:40.

[2] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Alastair Green, Jan Hidders, Bei Li, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Stefan Plantikow, Ognjen Savkovic, Michael Schmidt, Juan Sequeda, Slawek Staworko, Dominik Tomaszuk, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Dusan Zivkovic. 2022. PG-Schema: Schemas for Property Graphs. *CoRR* abs/2211.10962 (2022).

[3] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, and Dominik Tomaszuk. 2021. PG-Keys: Keys for Property Graphs. In *SIGMOD Conference*. ACM, 2423–2436.

[4] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. 2009. The DL-Lite Family and Relations. *J. Artif. Intell. Res.* 36 (2009), 1–69.

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

[6] Pablo Barceló Baeza. 2013. Querying graph databases. In *PODS*. ACM, 175–188.

[7] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. 2005. Reasoning on UML class diagrams. *Artif. Intell.* 168, 1-2 (2005), 70–118.

[8] Iovka Boneva, Benoît Groz, Jan Hidders, Filip Murlak, and Slawek Staworko. 2023. Static Analysis of Graph Database Transformations. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Seattle, WA, USA) *(PODS '23)*. Association for Computing Machinery, New York, NY, USA, 251–261. https://doi.org/10.1145/3584372.3588654

[9] Angela Bonifati, Wim Martens, and Thomas Timm. 2019. Navigating the Maze of Wikidata Query Logs. In *WWW*. ACM, 127–138.

[10] Angela Bonifati, Wim Martens, and Thomas Timm. 2020. An analytical study of large SPARQL query logs. *VLDB J.* 29, 2-3 (2020), 655–679.

[11] Pierre Bourhis and Carsten Lutz. 2016. Containment in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics. In *KR*. AAAI Press, 207–216.

[12] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the Decidability of Query Containment under Constraints. In *PODS*. ACM Press, 149–158.

[13] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *KR*. Morgan Kaufmann, 176–185.

[14] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2002. Rewriting of Regular Expressions and Regular Path Queries. *J. Comput. Syst. Sci.* 64, 3 (2002), 443–465.

[15] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. 1998. Description Logics for Conceptual Data Modeling. In *Logics for Databases and Information Systems*. Kluwer, 229–263.

[16] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. 1999. Unifying Class-Based Representation Formalisms. *J. Artif. Intell. Res.* 11 (1999), 199–240.

[17] Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. 2011. Containment of Regular Path Queries under Description Logic Constraints. In *IJCAI*. IJCAI/AAAI, 805–812.

[18] Daniel Danielski and Emanuel Kieronski. 2019. Finite Satisfiability of Unary Negation Fragment with Transitivity. In *MFCS (LIPIcs, Vol. 138)*. 17:1–17:15.

[19] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Oskar van Rest, Hannes Voigt, Domagoj Vrgoc, Mingxi Wu, and Fred Zemke. 2022. Graph Pattern Matching in GQL and SQL/PGQ. In *SIGMOD Conference*. ACM, 2246–2258.

[20] Alin Deutsch and Val Tannen. 2001. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *DBPL (Lecture Notes in Computer Science, Vol. 2397)*. Springer, 21–39.

[21] Diego Figueira. 2020. Containment of UC2RPQ: The Hard and Easy Cases. In *ICDT (LIPIcs, Vol. 155)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:18.

[22] Diego Figueira, Adwait Godbole, Shankara Narayanan Krishna, Wim Martens, Matthias Niewerth, and Tina Trautner. 2020. Containment of Simple Conjunctive Regular Path Queries. In *KR*. 371–380.

[23] Daniela Florescu, Alon Y. Levy, and Dan Suciu. 1998. Query Containment for Conjunctive Queries with Regular Expressions. In *PODS*. ACM Press, 139–148.

[24] Nadime Francis, Amélie Gheerbrant, Paolo Guagliardo, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Liat Peterfreund, Alexandra Rogova, and Domagoj Vrgoc. 2023. A Researcher's Digest of GQL (Invited Talk). In *ICDT (LIPIcs, Vol. 255)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1:1–1:22.

[25] Jakub Galgonek, Tomáš Hurt, Vendula Michlíková, Petr Onderka, Jan Schwarz, and Jiří Vondrášek. 2016. Advanced SPARQL querying in small molecule databases. *Journal of Cheminformatics* 8, 1 (2016), 31. https://doi.org/10.1186/s13321-016-0144-4

[26] Abraham Ginzburg. 1968. *Algebraic Theory of Automata.* Academic Press.

[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 *KR*. 424–433.

[28] Víctor Gutiérrez-Basulto, Albert Gutowski, Yazmín Ibáñez-García, 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 2022, Haifa, Israel. July 31 - August 5, 2022*, Gabriele Kern-Isberner, Gerhard Lakemeyer, and Thomas Meyer (Eds.). https://proceedings.kr.org/2022/19/

[29] Víctor Gutiérrez-Basulto, Yazmín Ibáñez-García, Jean Christoph Jung, and Filip Murlak. 2023. Answering regular path queries mediated by unrestricted SQ ontologies. *Artif. Intell.* 314 (2023), 103808.

[30] David Harel, Jerzy Tiuryn, and Dexter Kozen. 2000. *Dynamic Logic.* MIT Press, Cambridge, MA, USA.

[31] 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

[32] Artem Lysenko, Irina A. Roznovăţ, Mansoor Saqi, Alexander Mazein, Christopher J. Rawlings, and Charles Auffray. 2016. Representing and querying disease networks using graph databases. *BioData Mining* 9, 1 (2016), 23.

[33] Martin Otto. 2010. Highly Acyclic Groups, Hypergraph Covers and the Guarded Fragment. In *LICS*. IEEE Computer Society, 11–20.

[34] Vaughan R. Pratt. 1979. Models of Program Logics. In *FOCS*. IEEE Computer Society, 115–122.

[35] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71.

A PROOFS FOR SECTION 5: ENTAILMENT OF ONE-WAY QUERIES

A.1 Proof of Lemma 5.3

For the left-to-right implication, consider graph G that realizes τ, satisfies $\mathcal{T}$, and refutes Q. Let $\Sigma_\mathcal{T}$ be the set of role names used in $\mathcal{T}$. For a node u in G and $r \in \Sigma_\mathcal{T}$, let G_u^r be the pointed graph obtained by taking a fresh copy of G, adding label $C_\rightarrow$ to all nodes and taking the copy of u for the distinguished node. Pointed graph $G_u^{r^-}$ is defined similarly, but all nodes are labelled with $C_\leftarrow$. Consider a frame with a node f_u^r labelled with G_u^r and node $f_u^{r^-}$ labelled with $G_u^{r^-}$ for each node u in G and role $r \in \Sigma_\mathcal{T}$. Whenever there is an r-edge from u to v in G with $r \in \Sigma_\mathcal{T}$, add an edge from $f_w^{s^-}$ to f_v^r with label $(u_w^{s^-}, r)$ and from f_w^s to $f_u^{r^-}$ with label (v_w^s, r^-) for every node w in G and every $s \in \Sigma_\mathcal{T}$, where $u_w^{s^-}$ is the copy of u in $G_w^{s^-}$ and v_w^s is the copy of v in G_w^s. The resulting concrete frame F is alternating by construction and obviously realizes τ. Up to labels $C_\leftarrow$ and $C_\rightarrow$, every component of F is isomorphic to G, and every connector of F is isomorphic to the one-step unravelling of a subgraph of G formed by a node and all its successors, or a node and all its predecessors, over role names from $\Sigma_\mathcal{T}$. It follows that all components and connectors refute Q, which means that F refutes Q. It also follows that all components satisfy $\mathcal{T}$, and in each connector the distinguished node satisfies either $\{\top \sqsubseteq C_\rightarrow\} \cup \mathcal{T}_\leftarrow$ or $\{\top \sqsubseteq C_\leftarrow\} \cup \mathcal{T}_\rightarrow$; this means that F satisfies $\mathcal{T}$.

Next, we turn the concrete alternating frame F into an abstract alternating frame F' that represents F. Let $\Gamma_{F'}$ be the set of labels used in τ, $\mathcal{T}$, or $\widehat{Q}$, plus $C_\rightarrow$ (recall that $C_\leftarrow = \bar{C}_\rightarrow$). We modify F as follows. Consider a frame node f in F. For each maximal type σ over $\Gamma_{F'}$ realized in G_f, choose an arbitrary node v_σ of type σ in G_f. Adjust edges originating in f to ensure that $G_{f,\sigma}$ is isomorphic to G_{f,v_σ}: edges with labels of the form (v_σ, r) should be relabelled with (σ, r), the remaining ones should be removed. Let v_f be the distinguished node in G_f. Relabel f with $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$ where

- $\tau_f = \{A \in \Gamma_{F'}^\pm \mid v_f \text{ has label } A \text{ in } G_f\}$;
- $\mathcal{T}_f$ is $\mathcal{T}_\rightarrow$ if v_f has label $C_\rightarrow$ in G_f, and $\mathcal{T}_\leftarrow$ if it has label $C_\leftarrow$;
- Θ_f is the set of maximal types over $\Gamma_{F'}$ realized in G_f;
- $Q_f = \widehat{Q}$.

The resulting abstract alternating frame F' is obviously productive: G_f is a witness for $(\tau_f, \mathcal{T}_f, \Theta_f, Q_f)$. Also, F' realizes τ, satsfies $\mathcal{T}$, and refutes Q, because so does F.

For the right-to-left implication, take such a productive abstract frame and consider some concrete frame F it represents. As we have argued, because F is alternating, each RPQ has span at most 1 in F. Applying Lemma 4.3, we obtain a frame F' that actually refutes Q and preserves all local properties of F. As these include being alternating, satisfying $\mathcal{T}$, and realizing τ, by Lemma 5.2 it follows that $G_{F'}$ realizes τ, satisfies $\mathcal{T}$, and refutes Q.

A.2 Finding witnesses guaranteed by Lemma 5.3

In order to prove Theorem 5.1, it remains to show that the existence of a productive alternating abstract frame that realizes type τ, satisfies TBox $\mathcal{T}$, and weakly refutes query Q, can be decided in 2EXPTIME. Whether an abstract frame satisfies all these conditions depends exclusively on the set of isomorphism types of components and connectors it is build from. And of course it has to be a frame: for every type allowed in a component, we need a connector with the distinguished node of this type, and conversely, for every non-distinguished node in a connector, we need a component with distinguished node of the same type.

Keeping in mind that $C_\leftarrow = \bar{C}_\rightarrow$, we can restrict our attention to abstract frames involving only types over Γ_0, defined as the set of concept names containing $C_\rightarrow$ and all concept names used in τ, $\mathcal{T}$, and $\widehat{Q}$. Consider the set $\Psi_{\mathcal{T},Q}$ of all maximal types over Γ_0 realized in concrete alternating frames that satisfy $\mathcal{T}$ and weakly refute Q. The set $\Psi_{\mathcal{T},Q}$ coincides with the set of maximal types over Γ_0

that are realized in a component of such a frame, not necessarily in the distinguished node. Indeed, for a type σ realized by a node v in a component G_f, we can add a fresh frame node e labelled with G_f with the distinguished node changed to v, copying all the outgoing edges from f, and the result will still be an alternating concrete frame that satisfies $\mathcal{T}$ and weakly refutes Q.

For a set Ψ of maximal types over Γ_0, let $\Psi_\rightarrow$ and $\Psi_\leftarrow$ denote the sets of types from Ψ that contain $C_\rightarrow$ and $C_\leftarrow$, respectively. The set $\Psi_{\mathcal{T},Q}$ is the greatest (wrt. inclusion) set Ψ of maximal types over Γ_0 such that for each $\sigma \in \Psi$:

- if $\sigma \ni C_\rightarrow$, then the component $(\sigma, \mathcal{T}_\rightarrow, \Psi_\rightarrow, \widehat{Q})$ is productive, and there is a directed connector refuting Q whose distinguished node is of type σ and satisfies $\mathcal{T}_\leftarrow$, and whose non-distinguished nodes are of types from $\Psi_\leftarrow$;
- if $\sigma \ni C_\leftarrow$, then the component $(\sigma, \mathcal{T}_\leftarrow, \Psi_\leftarrow, \widehat{Q})$ is productive, and there is a directed connector refuting Q whose distinguished node is of type σ and satisfies $\mathcal{T}_\rightarrow$, and whose non-distinguished nodes are of types from $\Psi_\rightarrow$.

Note that, for a fixed σ, the conditions are monotone w.r.t. Ψ: that is, if they hold for Ψ, they also hold for each $\Psi' \supseteq \Psi$. Consequently, we can compute $\Psi_{\mathcal{T},Q}$ by a simple greatest fixed point procedure. For $i = 0, 1, 2, \ldots$, we iteratively compute sets Ψ_i of maximal types σ over Γ_0 that satisfy the conditions above with $\Psi = \Psi_{i-1}$, starting from the set Ψ_0 of all maximal types over Γ_0. We stop when $\Psi_{i+1} = \Psi_i$ and let $\Psi_{\mathcal{T},Q} = \Psi_i$. Then, it suffices to check whether $\Psi_{\mathcal{T},Q}$ contains a type σ such that $\sigma \supseteq \tau$.

For a given σ and $\Psi = \Psi_\rightarrow \cup \Psi_\leftarrow$, the conditions can be verified in 2EXPTIME. Indeed, testing if an abstract component is productive amounts to solving the corresponding instance of finite entailment; it either is an instance with $\mathcal{ALC}$ TBox, or it can be easily turned into one, as we have explained in the main part of the paper. (In the latter case, we also need to adjust the query: reverse the transitions in the underlying semiautomaton and replace each atom $\mathcal{A}_{s,s'}(y, y')$ with $\mathcal{A}_{s',s}(y', y)$.) Moreover, we can eliminate tests from $\widehat{Q}$, by encoding node types in the labels of the outgoing edges. Hence, we can apply the algorithm from [28]. While this algorithm has doubly exponential complexity, it is in fact only doubly exponential in the maximal size of any involved CRPQ. Each of our instances involves a union of exponentially many CRPQs of polynomial size, and a TBox of exponential size (due to the elimination of tests). It follows that the overall complexity of each productivity test is still doubly exponential. The existence of a suitable connector for σ and $\Psi = \Psi_\rightarrow \cup \Psi_\leftarrow$ can also be decided in 2EXPTIME. Indeed, observe that it suffices to consider connectors with at most one non-distinguished node for each participation constraint in $\mathcal{T}$. There are doubly exponentially many such connectors (despite the number of available types already being doubly exponential). Checking that a given connector refutes Q amounts to evaluating $\widehat{Q}$ over the connector, which can be done in time $O(|\widehat{Q}| \cdot m^n \cdot n \cdot m \cdot k)$, where n is the maximal size of a CRPQ in $\widehat{Q}$ (linear in the maximal size of a CRPQ in Q), m is size of the connector, and k is the size of the semiautomaton underlying Q. Checking the remaining conditions is straightforward. Hence, each iteration of the greatest fixed point procedure takes doubly exponential time. The number of iterations is also at most doubly exponential, because we begin from a doubly exponential set of types, and in each iteration at least one type is eliminated.

B PROOFS FOR SECTION 6: ENTAILMENT OF TWO-WAY QUERIES

B.1 No roles

We need to solve the following problem: given a type τ, a TBox $\mathcal{T}$ mentioning no roles, a set Θ of types, a connected simple UC2RPQ Q, and $\Sigma_1 \subseteq \Sigma$, decide if τ is realized in a graph that satisfies $\mathcal{T}$, has only nodes of types from Θ, and does not satisfy $\widehat{Q}$ mod Σ_0. Because $\mathcal{T}$ does not mention any roles, we can restrict our attention to graphs consisting of a single isolated node. Moreover, we can assume that the graph uses only node labels mentioned in τ, $\mathcal{T}$, Θ, and $\widehat{Q}$. As the number of labels is exponential, the number of such graphs is doubly exponential. Checking if one such graph is a witness we are looking for can be done in time exponential in the size of the input.

B.2 Proof of Lemma 6.3

Before proving Lemma 6.3, we need to lift Lemma 4.1 to refuting modulo Σ_0 reachability.

Lemma B.1. *If a concrete frame F is a tree and weakly refutes a connected simple UC2RPQ Q modulo (Σ_0, Σ_0)-reachability, then G_F refutes Q modulo Σ_0-reachability.*

Proof. Just like for $\widehat{Q}$, we can prove that $\widehat{Q}$ mod Σ_0 is factorized. Given that, the claim follows immediately from Lemma 4.2 for $P = \widehat{Q}$ mod Σ_0. □

Proof of Lemma 6.3. For the left-to-right implication, consider a graph G that realizes τ, satisfies $\mathcal{T}$, has only nodes of types from Θ, and refutes Q modulo Σ_0-reachability. Without loss of generality we can assume that G only uses edge labels from $\Sigma_{\mathcal{T}}$.

For each node v in G let G_v be a pointed graph obtained by taking a fresh copy of the strongly connected component of G that contains v, with v as the distinguished node. Consider a graph F_0 with a node f_v labelled with G_v for each node v in G. Whenever there is an r-edge from u to v, add an edge from f_w to f_v with label (u_w, r) for every w whose SCC in G contains u, with u_w being the copy of u in G_w. Graph F_0 need not be a tree. In fact, it may not even be a concrete frame, because it may contain parallel edges that share the first component of the label. However, because F_0 is acyclic, we can turn it into a tree by unravelling it from any component whose distinguished node is of type τ (acyclicity ensures that the unravelling is finite), and adjusting the labels so that all components are disjoint. The resulting graph F is a concrete frame and also a tree. It obviously realizes τ and all its components only contain nodes of types in Θ. In each component of F we add labels $C_{n,r,D}$ in the unique way that ensures that F satisfies $\mathcal{T}$. As the added labels are fresh, this does not affect Θ.

Up to labels $C_{n,r,D}$, every component of F is isomorphic to an SCC in G, and every connector of F is isomorphic to the one-step unravelling of a subgraph of G formed by a node and all its successors. It follows that all components and connectors refute Q modulo Σ_0-reachability. Moreover, because each component of F is strongly connected via edges with labels from $\Sigma_{\mathcal{T}} \subseteq \Sigma_0$, it actually refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability. That is, F refutes Q modulo $(\Sigma_{\mathcal{T}}, \Sigma_0)$-reachability.

To conclude the proof of the left-to-right implication, we turn F into a suitable abstract frame F' as in the proof of Lemma 5.3, with the following differences. For $\Gamma_{F'}$ we take the set of node labels used in τ, $\mathcal{T}$, Θ, and $\widehat{Q}$, plus all node labels of the form $C_{n,r,D}$ used in $\mathcal{T}_{=}$ and $\mathcal{T}_{+}$. For each frame node f, we let $\mathcal{T}_f = \mathcal{T}_{=}$ and $Q_f = \widehat{Q}$ mod $\Sigma_{\mathcal{T}}$.

For right-to-left implication, take such a productive abstract frame and consider some concrete frame F it represents. Obviously, G_F realizes τ and contains only nodes of types from Θ. Because F refutes Q modulo $(\Sigma_{\mathcal{T}}, \Sigma_0)$-reachability and $\Sigma_{\mathcal{T}} \subseteq \Sigma_0$, F also refutes Q modulo (Σ_0, Σ_0)-reachability. By Lemma B.1, G_F refutes Q modulo Σ_0-reachability. By Lemma 6.2, $G_F \models \mathcal{T}$. This completes the proof of the right-to-left implication. □

B.3 Finding witnesses guaranteed by Lemma 6.3

It remains to test if an abstract frame satisfying the conditions from Lemma 6.3 exists. This time we use a least-fixed-point algorithm. We compute the set $\Psi_{\mathcal{T},\Theta,Q}$ of types of distinguished elements in components of abstract frames satisfying the conditions of Lemma 6.3. This set coincides with the set of all unary types that occur in interpretations represented by such abstract frames. We compute it iteratively: the set Ψ_1 of types appearing in leaves of such abstract frames; the set Ψ_2 of types appearing in leaves or their parents, etc.

Let Γ_0 be the set of concept names in τ, $\mathcal{T}$, $\widehat{Q}$, Θ, and $C_{\mathcal{T}}$. Let $\Psi_0 = \emptyset$. For $i > 0$, Ψ'_i is defined as the set of maximal types σ over Γ_0 such that there σ contains a type from Θ and there is a connector with the distinguished element of type σ, non-distinguished elements of types from Ψ_{i-1}, and satisfying all requirements for connectors in witnessing abstract frames from Lemma 6.3. It suffices to look for connectors with at most N distinguished elements per participation constraint in $\mathcal{T}$. Like in Section A.2, whether such a connector exists can be decided in 2EXPTIME. The set Ψ_i is defined as the set of types σ such that the abstract component $(\sigma, \mathcal{T}_=, \Psi'_i, \widehat{Q} \bmod \Sigma_{\mathcal{T}})$ is productive. That is, we must ensure that σ is realizable in a finite graph that satisfies $\mathcal{T}_=$, respects Ψ'_i, and refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability atoms, which is precisely the problem we want to reduce to. We compute Ψ_i iteratively for $i = 1, 2, \ldots$ until $\Psi_i = \Psi_{i+1} = \Psi$. Then, it suffices to check whether the resulting set Ψ contains a type σ such that $\sigma \supseteq \tau$.

B.4 Proof of Lemma 6.4

Each simple 2RPQ is either of the form $r(x, y)$ or $R^*(x, y)$ for a set $R \subseteq \Sigma^\pm$. A 2RPQ of the form $r(x, y)$ has span at most 1. Recall that the balance of an undirected path is the difference between the number of edges traversed forward and backward by the path. Note that for very label (v, r) of an edge in a role-alternating frame, r is a role name, as each connector is role-directed. If a simple 2RPQ $R^*(x, y)$ is not a $\Sigma_{\mathcal{T}}$-reachability atom, there exist role names $r, s \in \Sigma_{\mathcal{T}}$ such that $r \notin R$ and $s^- \notin R$ (since otherwise $\Sigma_{\mathcal{T}} \subseteq R$ or $\Sigma_{\mathcal{T}}^- \subseteq R$). Let σ be an undirected path in a role-alternating frame F induced by a path witnessing such a 2RPQ in G_F. Consider the sequence of roles in the labels of edges traversed by σ, putting t if an edge with label (v, t) is traversed forward, or t^- if it is traversed backward. Two consecutive elements in the sequence are always one of (r_k, r_{k+1}), (r_{k+1}^-, r_k^-), (r_k, r_k^-) or (r_k^-, r_k) for some $k \in \{1, \ldots, n\}$. The latter two pairs do not contribute to the balance of the path; let us iteratively remove all such pairs from the sequence until none are left. We remain with a sequence of only role names, or only inverse roles; the length of the sequence is exactly the absolute value of balance of σ. Since the role name r and inverse role s^- are not mentioned by the 2RPQ, the length of the sequence is bounded by $|\Sigma_{\mathcal{T}}| - 1$. We can apply the same reasoning to every infix of σ, proving that the span of σ is at most $|\Sigma_{\mathcal{T}}| - 1$.

B.5 Proof of Lemma 6.5

Before proving Lemma 6.5, we need to generalize Lemma 4.3 to refuting modulo Σ_0-reachability.

Lemma B.2. *Let F be a concrete frame all of whose nodes are reachable from a single node*[1] *and let Q be a connected simple UC2RPQ. If F weakly refutes Q modulo (Σ_0, Σ_0)-reachability and all 2RPQs in $\widehat{Q}$ that are not Σ_0-reachability atoms have bounded span in F, then there is a frame locally isomorphic to F that actually refutes Q modulo Σ_0-reachability.*

Proof. There exists $k > 0$ such that, for each 2RPQ p in $\widehat{Q}$ that is not a Σ_0-reachability atom, the span of p in F is at most k. Let $m = \max\{|q| : q \in Q\}$.

A locally isomorphic frame that actually refutes Q is obtained by converting the generalized frame $\mathrm{Coil}(F, km + d)$ to a frame, where d is the number of nodes in F. Let f_0 be a node in F from which all nodes in F are reachable. By Property 2, there exists a subgraph T of $\mathrm{Coil}(F, km + d)$ isomorphic with $\mathrm{Unravel}(F, km + d - 1, f_0)$. Since every node in F is reachable from f_0 by a simple path of length at most $d - 1$, the tree T contains a subtree isomorphic to $\mathrm{Unravel}(F, km, f)$ for each node f in F.

[1]The lemma holds also if F is only assumed to be connected, but the present statement is sufficient and a bit easier to prove.

Every match of $\widehat{Q}$ mod Σ_0 in Coil$(F, km+d)$ is a match of some C2RPQ $q \in \widehat{Q}$ mod Σ_0, which can be seen as a collection of matches of connected components of q. Every match of such a connected component, by Lemma 2, visits at most km levels; by Property 3, it maps to Unravel(F, km, v) for some node v in F. Thus, every match of $\widehat{Q}$ mod Σ_0 in Coil$(F, km + d)$ maps homomorphically to T. By converting T to a frame we obtain a tree-shaped frame that weakly refutes Q modulo (Σ_0, Σ_0)-reachability; by Lemma B.1, $\tilde{G_T}$ refutes Q modulo Σ_0-reachability, which proves that there is no match of $\widehat{Q}$ mod Σ_0 in Coil$(F, km + d)$. □

Proof of Lemma 6.5. For the left-to-right implication, consider a graph G that realizes τ, satisfies $\mathcal{T}$, has only nodes of types from Θ, and refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability. Without loss of generality we can assume that G only uses edge labels from $\Sigma_{\mathcal{T}}$.

Suppose first that $|\Sigma_{\mathcal{T}}| > 1$. Let $r_1, r_2, \ldots, r_m$ be an enumeration of $\Sigma_{\mathcal{T}}$ and let $r_{m+1} = r_1$. Because $m > 1$, $r_i \neq r_{i+1}$. For each node v in G and each r_i, let G_{v,r_i} be a pointed graph obtained by taking a fresh copy of G with all r_i-edges removed, label C_{r_i} added to each node, and v chosen for the distinguished node. Consider a graph F with a node f_{v,r_i} labelled with G_{v,r_i} for each node v in G and each r_i. Whenever there is an r_i-edge from u to v, add an edge from f_{w,r_i} to $f_{v,r_{i+1}}$ with label (u_{w,r_i}, r_i) for every w in G, where u_{w,r_i} is the copy of u in G_{w,r_i}. Because $r_{i+1} \neq r_i$, graph F contains no self-loops. By construction, F does not contain parallel edges with the same first component in the label. Hence, F is a concrete frame.

Label each component in F with $C_{n,r,D}$ in the unique way that ensures that F is role-alternating and satisfies $\mathcal{T}$. Frame F obviously realizes τ and all its components have only nodes of types from Θ (node labels $C_{n,r,D}$ are fresh, so they do not matter for Θ). Up to labels $C_{n,r,D}$ and C_r, every component of F is isomorphic to a subgraph of G, and every connector of F is isomorphic to the one-step unravelling of a subgraph of G formed by a node and all its successors. It follows that all components and connectors refute Q modulo $\Sigma_{\mathcal{T}}$-reachability.

To conclude the proof of the left-to-right implication in the case when $|\Sigma_{\mathcal{T}}| > 1$, we turn F into a suitable abstract frame F' as in the proof of Lemma 6.3, with the following differences. For $\Gamma_{F'}$ we take the set of node labels used in τ, $\mathcal{T}$, Θ, and $\widehat{Q}$, plus all node labels $C_{n,r,D} \in \Gamma_{\mathcal{T}}$ and all node labels of the form C_r with $r \in \Sigma_{\mathcal{T}}$. For each frame node f such that the distinguished node in G_f has label C_r, we let Θ_f be the set of all maximal types over $\Gamma_{F'}$ that contain some type from Θ and include C_r, all $C_{0,r,D} \in \Gamma_{\mathcal{T}}$, and all $\bar{C}_s$ with $s \in \Sigma_{\mathcal{T}} \setminus \{r\}$.

Suppose now that $\Sigma_{\mathcal{T}} = \{r\}$. This time, for each v in G, and each $i \in \{0, 1\}$, we let G^i_v be a pointed graph consisting of a single isolated fresh node v^i, with labels inherited from v, plus an additional label C_r. We let F be a a graph with nodes f^0_v and f^1_v for each node v in G, labelled with G^0_v and G^1_v, respectively. Whenever there is an r-edge from u to v in G, we add an edge from f^0_u to f^1_v labelled with (u^0, r) and an edge from f^1_u to f^0_v labelled with (u^1, r). Thanks to alternating between the two sets of components, F contains no self-loops. It follows easily that F is a concrete frame. From there, we continue like in the case with $|\Sigma_{\mathcal{T}}| > 1$.

For the right-to-left implication, take such a productive abstract frame and consider a concrete frame F it represents. Frame F realizes τ, satisfies $\mathcal{T}$, weakly refutes Q modulo $\Sigma_{\mathcal{T}}$-reachability, and its components only contain nodes of types from Θ. Lemma 6.4 allows us to apply Lemma B.2 to the frame obtained from F by restricting to frame nodes reachable from an arbitrarily chosen frame node f such that the distinguished element of G_f is of type τ. The graph represented by the concrete frame resulting from Lemma B.2 has all the properties we need. □

B.6 Finding witnesses guaranteed by Lemma 6.5

The algorithm is very similar to the one described in Section A.2. Let Γ_0 be the set containing all concept names used in τ, $\mathcal{T}$, Θ, and $\widehat{Q}$, as well as those from $\Gamma_{\mathcal{T}}$ and all C_r with $r \in \Sigma_{\mathcal{T}}$.

We aim to compute the set $\Psi_{\mathcal{T},\Theta,Q}$ of maximal types over Γ_0 that are realized in abstract frames with properties specified in Lemma 6.5. For a set Ψ of types over Γ_0 and a role name $r \in \Sigma_{\mathcal{T}}$, let $\Psi_r = \left\{\sigma \in \Psi \;\middle|\; C_r \in \sigma\right\}$. Let $r_1, r_2, \ldots, r_m$ be an enumeration of $\Sigma_{\mathcal{T}}$. The set $\Psi_{\mathcal{T},\Theta,Q}$ is the greatest set Ψ of maximal types over Γ_0 such that

$$\Psi = \Psi_{r_1} \cup \Psi_{r_2} \cup \cdots \cup \Psi_{r_m}$$

and for each $i \in \{1, 2, \ldots, m\}$ and $\sigma \in \Psi_{r_i}$,

- σ contains all concept names $C_{0,r_i,D} \in \Gamma_{\mathcal{T}}$ and $\bar{C}_{r_j}$ with $j \neq i$;
- $\sigma \supseteq \sigma'$ for some $\sigma' \in \Theta$;
- abstract component $(\sigma, \mathcal{T}_=, \Psi_{r_i}, \widehat{Q} \bmod \Sigma_{\mathcal{T}})$ is productive;
- there is a connector with distinguished element of type σ and non-distinguished elements of types from $\Psi_{r_{i+1}}$, that satisfies all properties required for connectors of witnessing frames in Lemma 6.5.

We can compute it using a greatest fixed-point procedure, as in Section A.2, starting from the set of all maximal types over Γ_0 that, for some $i \in \{1, 2, \ldots, m\}$, contain C_{r_i}, all $C_{0,r_i,D} \in \Gamma_{\mathcal{T}}$, and all $\bar{C}_{r_j}$ with $j \neq i$. Checking if there exists a suitable connector is done exactly as in Section B.3. Let us look closer at testing productivity of $(\sigma, \mathcal{T}_=, \Psi_{r_i}, \widehat{Q} \bmod \Sigma_{\mathcal{T}})$. Because it is always guaranteed that all types in Ψ_{r_i} contain all $C_{0,r_i,D} \in \Gamma_{\mathcal{T}}$, we can drop from $\mathcal{T}_=$ all CIs that involve role r_i. This way we reduce the productivity test to finite entailment modulo $\Sigma_{\mathcal{T}}$, but the involved TBox has one role name fewer. This allows us to use a recursive call to our decision procedure.

B.7 Overall complexity

Overall, our recursive procedure has the depth twice the number of roles used in the original TBox. At each depth of recursion we perform doubly exponential computation and make doubly exponentially many recursive calls. When passing from depth 0 to depth 1 of the recursion, the original TBox $\mathcal{T}$ is replaced with $\mathcal{T}_=$, which has exponential size and introduces exponentially many fresh concept names $\Gamma_{\mathcal{T}}$. Deeper in the recursion, we re-apply the same construction to $\mathcal{T}_=$. What this does is replace previously introduced concepts $C_{n,r,D}$ with their fresh copies, effectively giving an identical TBox, just over a different set of concept names. The construction also produces the TBox $(\mathcal{T}_=)_+$, that is still of exponential size; this TBox is only used to find suitable connectors, and is not passed down the recursion. The new rounds of concepts $C_{n,r,D}$ introduced at each level of the reduction do accumulate in the type τ to realize and set Θ of allowed types, but the total number of added concept names is still only exponential in the size of the original TBox. The algorithm for the case without roles, described in Section B.1, can handle this within 2EXPTIME.

Received June 2023; revised August 2023; accepted September 2023