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:

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...
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 ($Q$) and see edges backwards ($I$) [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 $S$ 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 $q_1(x, y) = (\text{Owns} \cdot \text{Earns} \cdot \text{Partner} \cdot \text{Owns}^*) (x, y)$ and $q_2(x, y) = (\text{Owns-Earns-Partner}) (x, z) \land \text{RetailCompany}(z) \land \text{Owns}^*(z, y)$; we use $\cdot$ for concatenation
and * for unbounded iteration. Without schema, \( q_2 \) is contained in \( q_1 \), but not conversely; modulo \( S \), \( q_1 \) 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. UC2RPQs 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 \( 2\text{EXPTIME} \); 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 \emph{coil}.

2 PRELIMINARIES

Graphs. We fix a recursively enumerable set \( \Gamma \) of node labels, and an recursively enumerable set \( \Sigma \) 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 \( \Gamma \) and binary relation symbols \( \Sigma \). That is, a graph \( G \) is a pair \((\text{dom}(G), G)\) where \( \text{dom}(G) \) is the set of nodes of \( G \) and the function \( \cdot^G \) maps each \( A \in \Gamma \) to a set \( A^G \subseteq \text{dom}(G) \) and each \( r \in \Sigma \) to a binary relation \( r^G \subseteq \text{dom}(G) \times \text{dom}(G) \). Graph \( G \) is finite if \( \text{dom}(G) \) 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 \( \text{dom}(G) \subseteq \text{dom}(G') \), \( A^G \subseteq A^{G'} \) 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 \( h : G \rightarrow G' \), is a function \( h : \text{dom}(G) \rightarrow \text{dom}(G') \) such that \( u \in A^G \iff h(u) \in A^{G'} \) and \( (u,v) \in r^G \) implies \( (h(u),h(v)) \in r^{G'} \) for all \( u,v \in \text{dom}(G) \), \( A \in \Gamma \), 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 \( \bar{A}^G = \text{dom}(G) \setminus A^G \). We write \( \Gamma_0^- = \{ A | A \in \Gamma_0 \} \), \( \Gamma_0^+ = \Gamma_0 \cup \Gamma_0^- \) for every \( \Gamma_0 \subseteq \Gamma \). A type is a subset of \( \Gamma^* \) that contains at most one of \( A \) and \( \bar{A} \) for all \( A \in \Gamma \). A type over
We say that \( r, r \) we mean binary atoms of \( C(2)RPQs \). A query is test-free where \( \pi_n \) for some \( \Theta \) of types if each node in \( G \) is of some type from \( \Theta \). We use \( r^- \) for inverse edges and let \( (r^-)^G = \{ (u, v) \mid (v, u) \in r^G \} \), \( \Sigma^-_0 = \{ r^- \mid r \in \Sigma_0 \} \), \( \Sigma^\pm_0 = \Sigma_0 \cup \Sigma^-_0 \) 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

\[
q = A_1(x_1) \land \cdots \land A_k(x_k) \land \varphi_1(y_1, z_1) \land \cdots \land \varphi_m(y_m, z_m),
\]

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 \( \text{var}(q) \) for the set of variables used in \( q \). A match of \( q \) in a graph \( G \) is a function \( \pi : \text{var}(q) \to \text{dom}(G) \) 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 \( v_0, \ldots, v_n \in \text{dom}(G) \) 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 \( u_{i-1} = v_i = (\ell_i)^G \);
3. the word \( \ell_1 \ldots \ell_n \) matches the regular expression \( \varphi \).

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 \( \Sigma^- \) in regular expressions. By (two-way) regular path queries, abbreviated as (2)RPQs, we mean binary atoms of (C)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 UC2RPQ \( Q \) can be effectively rewritten as a UC2RPQ \( Q' \) 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 \( \Gamma \) and \( \Sigma \) are called concept names and role names, respectively. \( \mathcal{ALCQI} \) allows building more complex concepts with the following grammar:

\[
C ::= \bot \mid A \mid C \cap C \mid \neg C \mid \exists^{\leq \alpha} r.C,
\]
where \( A \in \Gamma^\pm, r \in \Sigma^\pm, \) and \( n \in \mathbb{N} \). We extend the interpretation function \( ^G \) to complex concepts as follows:

\[
\bot^G = \emptyset, \quad (C_1 \cap C_2)^G = C_1^G \cap C_2^G, \quad (\neg C)^G = \text{dom}(G) \setminus C^G, \\
(\exists^{\leq n} r. C)^G = \{ u \in \text{dom}(G) \mid \exists^{\leq n} v. \ (u, v) \in r^G \land v \in C^G \}.
\]

We also use additional operators that are redundant but useful when defining fragments; for brevity we introduce them as syntactic sugar: \( \top \) where \( A \) earns \( \text{PremCC} \), and similarly for \( \perp \). For instance, \( S \) is the use of inverse role \( \text{dom} \) modulo schema \( \Gamma \), and the question to decide is whether \( G \models P \subseteq T Q \). We focus on the Boolean variant of the containment problem. Given queries \( P, Q \) and a TBox \( T \), we write

\[
P \subseteq_T Q
\]

and similarly for \( \text{Partner} \) and \( \text{owns} \). We must also ensure that entities do not overlap, unless explicitly allowed by generalization relationships; for instance, \( \text{Customer} \cap \text{CredCard} \subseteq \perp \) and \( \text{RwrdProg} \cap \text{Company} \subseteq \perp \). The use of inverse role \( \text{earns}^{-1} \) can be avoided, by flipping the CI to the contrapositive: \( \overline{\text{PremCC}} \subseteq \overline{\text{earns}.\text{RwrdProg}} \).

3 CONTAINMENT AND ENTAILMENT

We focus on the Boolean variant of the containment problem. Given queries \( P \) and \( Q \) and a TBox \( T \), we write

\[
P \subseteq_T Q
\]

if \( G \models P \) implies \( G \models Q \) for every finite graph \( G \) that satisfies \( T \). In the problem of containment modulo schema the input is \( P, Q, \) and \( T \), and the question to decide is whether \( P \subseteq_T Q \).
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 $A \subseteq \exists r.B$ and $A \subseteq \exists^n r.B$. We rely on a simple model property established in [17] in the context of unrestricted satisfiability of C2RPQs in the presence of an $\mathcal{ALCQI}$ 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 $h : H \rightarrow G$ such that for all $r \in \Sigma^+$ and $(u, v_1) \in r^H$, 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: $K \subseteq L$, $A \subseteq \forall r.B$, and $A \subseteq \exists^n r.B$. CIs of the form $K \subseteq L$ carry over from $G$ to $G_p$ because homomorphisms preserve types of nodes. CIs of the form $A \subseteq \forall r.B$ 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 $A \subseteq \exists^n r.B$ 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 $p \subseteq \mathcal{T}$ 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 $G, \mathcal{T} \models_{\text{fin}} Q$, if the above holds for every finite graph $G'$. The finite entailment problem is to decide if $G, \mathcal{T} \models_{\text{fin}} Q$ 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.
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 ALC [28].
2. UCQs with transitive atoms and ALCI or ALCQ [27].

The lower bound in (2) holds already for 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 ALCI or ALCQ,
2. simple UC2RPQs and ALCQ.

The 2EXPTIME lower bound of Theorem 3.3 carries over immediately to the containment of simple CRPQs modulo 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 ALCI or 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 ALCI or ALCQ TBox $\mathcal{T}$, $p \not\models_{\mathcal{T}} Q$ 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 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 $r \in \Sigma^+$ involved in a participation constraint in $T$, 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 $H \models \tau$: while one of the $r^+$-successors of the copy of $v$ is duplicated, we know that $r$ is involved in a participation constraint. If $T$ is an ALG TBox, the duplication cannot be detected, because the logic does not count. If $T$ is an 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. \hfill \Box

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 $H \models \tau$ and $H \not\models Q$ with local conditions over parts of $H$. We refer to this technique as factorizing the TBox $\tau$ and the query $Q$.

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

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

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

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

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

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

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

Example 3.6. Let $Q$ consist of a single CRPQ

$$A(x) \land r^+(x, y) \land B(y).$$

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

$$A(x) \land \hat{C_A}(x), \quad C_A(x) \land r^+(x, z) \land \hat{C_A}(z), \quad C_A(z) \land C_B(z), \quad \hat{C_B}(z) \land r^+(z, y) \land C_B(y), \quad \hat{C_B}(y) \land B(y).$$

Intuitively, $\hat{Q}$ 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 $\hat{H}$ 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, $\hat{H} \not\models \hat{Q}$. Now, let $\hat{H}$ be any graph obtained from $H$ by adding labels $C_A$ and $C_B$ in some nodes. Suppose that $\hat{H}$ satisfies $q_2 = C_A(x) \land r^+(x, z) \land C_A(z)$. 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 $\hat{H}$ satisfies $q_2$. For other disjuncts of $\hat{Q}$ the argument is either analogous or trivial. Hence, if $\hat{H} \models \hat{Q}$ then some part of $\hat{H}$ satisfies $\hat{Q}$. Note that complement node labels are crucial in this construction.

Lemma 3.7. Given a connected UC2RPQ $Q$, a connected UC2RPQ $\hat{Q}$ satisfying conditions (1) and (2) can be constructed in exponential time, while ensuring that $\hat{Q}$ is a union of C2RPQs of polynomial size, and if $Q$ is simple or one-way, so is $\hat{Q}$.
Proof. We begin by developing some auxiliary notions. A pointed C2RPQ \((q, x)\) is a connected C2RPQ \(q\) along with a distinguished variable \(x \in \text{var}(q)\). We say \((q, x)\) matches in graph \(G\) at node \(v\) if there is a match \(\pi\) 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 \(A_{s,t}(y, z)\) in \(q\) into three 2RPQs,

\[ A_{s,s'}(y, y'), A_{s',t'}(y', z'), A_{t',t}(z', z) \]

for fresh variables \(y'\) and \(z'\), corresponding to three segments of the path witnessing \(A_{s,t}(y, z)\) 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 \(A_{s,s'}(x', x')\) 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 \((A_{s,s'}(y, y), y)\).

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 \((q^0, x')\) 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 \(C_{A_{s,s'}(y, y), y}\) 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 \(x \in \text{var}(q)\). 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

\[ p' \land \widehat{C}_{p,y}(y'), \]

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 \(x \in \text{var}(q)\). 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).

\[ \square \]

Factorizing the TBox is easy. Let \(T_0\) be \(T\) with all participation constraints dropped. Because we assume that each peripheral part of \(H\) satisfies \(T\), we can replace the condition \(H \models T\) with

- the central part of \(H\) satisfies \(T_0\) and each of its nodes satisfies all participation constraints in \(T\) unless it is shared with a peripheral part.

The only delicate point is that CIs of the form \(A \subseteq \exists^<=r.B\) 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 \( \text{Tp}(\mathcal{T}, \hat{Q}) \) be the set of maximal types over labels used in \( \mathcal{T} \) and \( \hat{Q} \), realized in finite graphs that satisfy \( \mathcal{T} \) but not \( \hat{Q} \). We can now reformulate the criterion from Lemma 3.5 as follows: \( p \not\in \mathcal{T} Q \) iff there is a \(|p|\)-sparse graph \( H_0 \) such that \( H_0 \models p, H_0 \not\models T_0, H_0 \not\models \hat{Q} \), and each node violating a participation constraint from \( \mathcal{T} \) is of some type from \( \text{Tp}(\mathcal{T}, \hat{Q}) \), and has only one incident edge and, in the \( \mathcal{ALCQ} \) case, no outgoing edges. Given \( \text{Tp}(\mathcal{T}, \hat{Q}) \), we can adapt the automata-based argument behind Theorem 3.2 to decide if such \( H_0 \) exists. It remains to compute \( \text{Tp}(\mathcal{T}, \hat{Q}) \). For that we note that \( \tau \in \text{Tp}(\mathcal{T}, \hat{Q}) \) iff \( G_\tau, T \not\models \text{fin} \hat{Q} \), where \( G_\tau \) is a graph consisting of a single isolated node of type \( \tau \). That is, we can compute \( \text{Tp}(\mathcal{T}, \hat{Q}) \) by solving an instance of finite entailment for each maximal type over labels used in \( \mathcal{T} \) and \( \hat{Q} \). 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 \( \tau \in \text{Tp}(\mathcal{T}, \hat{Q}) \) can be realized in a finite graph \( G \) such that \( G \models \mathcal{T} \) and \( G \not\models \hat{Q} \).

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 \( v \in \text{dom}(G_f) \) and \( r \in \Sigma^\pm \). We assume that \( \text{dom}(G_f) \cap \text{dom}(G_e) = \emptyset \) 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 \( v \in \text{dom}(G_f) \) 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 \( r = s \) and \( s \in \Sigma^\pm \), 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 \( \tau \) if the distinguished node of some component of \( F \) is of type \( \tau \). Clearly, if \( F \) realizes type \( \tau \), 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 $G \not\models \hat{Q}$ (in particular, $G \not\models Q$). A concrete frame $F$ weakly refutes $Q$ if

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

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 $\hat{Q}$.

**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 $G_{f_{0},o}$ 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 $G_{f_0,o}$ 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. \(\square\)

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 $\text{Paths}(G, n)$ 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 $\text{Paths}(G, n, v)$ denote the set of paths in $\text{Paths}(G, n)$ that originate in $v$.

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

By the $n$-suffix of a path $\pi$ we mean the suffix of length $n$ of $\pi$ if $\pi$ has length at least $n$, and the whole path $\pi$ otherwise. For a graph $G$ and $n > 0$, $\text{Coil}(G, n)$ is the graph with nodes $\text{Paths}(G, n) \times \{0, \ldots, n\}$ and an edge $((\pi, \ell), (\pi', \ell'))$ whenever $\ell' \equiv \ell + 1 \pmod{(n + 1)}$ and $\pi'$ is the $n$-suffix of an extension of $\pi$ by one edge. The label of a node $(\pi, \ell)$ in $\text{Coil}(G, n)$ is inherited from the last node of $\pi$, and the label of an edge $((\pi, \ell), (\pi', \ell'))$ is inherited from the last edge of the path $\pi'$.
The coil has three key properties. Let \( h_G : \text{Coil}(G, n) \rightarrow G \) map a node \((\pi, \ell)\) in \(\text{Coil}(G, n)\) to the last node on \(\pi\).

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

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

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

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

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

Now, consider a node \(u' = (\pi', \ell')\) reachable from \(u = (\pi, \ell)\) by a path \(\sigma\) of length at most \(n\) in \(\text{Coil}(G, n)\). Then, by the second observation above, the length of \(\sigma\) is \(k = (\ell' - \ell) \mod{(n + 1)}\) and \(h_G(\sigma)\) is the \(k\)-suffix of \(\pi'\). In the light of the first observation above, this means that \(\sigma\) is a unique path of length at most \(n\) from \(u\) to \(u'\). It follows that the subgraph of \(\text{Coil}(G, n)\) 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 \(\text{Unravel}(G, n - 1, h_G(u))\) via the mapping \(u' \mapsto h_G(\sigma)\). \(\square\)

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

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

**Proof.** Since \(H\) does not visit all levels and is connected, there exists a unique level \(\ell_0\) such that \(H\) visits level \(\ell_0\), but does not visit level \((\ell_0 - 1) \mod{(n + 1)}\). For each node \((\pi, \ell)\) in \(H\), let us call the value \((\ell - \ell_0) \mod{(n + 1)}\) the **H-level** of \((\pi, \ell)\). The H-levels of nodes in \(H\) range from 0 to \(k - 1\). A mapping \(g\) from \((\pi, \ell)\) to the suffix of \(\pi\) of length equal to the H-level of \((\pi, \ell)\) is the required homomorphism. For each edge \(((\pi, \ell), (\pi', \ell'))\) in \(H\), \(\pi'\) is the \(n\)-suffix of the extension of \(\pi\) with one edge; since the H-level of \((\pi, \ell)\) 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 \(\text{Unravel}(G, k - 1, v)\). \(\square\)

**Avoiding UC2RPs.**

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 $\tilde{Q}$ 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 $\hat{h}_F : G_{F_n} \to G_F$ that preserves frame edges; that is, an edge in $G_{F_n}$, is a frame edge iff it is mapped by $h_F$ 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 $\sigma$ in $F_n$. Because $F_n$ differs from $\text{Coil}(F, n)$ only in the labelling of nodes and edges, we can view $\sigma$ as an undirected path in $\text{Coil}(F, n)$; this change of perspective does not affect the span of $\sigma$. Applying $h_F$ to the witnessing path in $G_{F_n}$, we obtain a witnessing path in $G_F$ such that the induced undirected path $\sigma'$ in $F$ can be obtained from $\sigma$ by applying $h_F$. As the span of a path is preserved by homomorphisms, it follows that $\sigma$ and $\sigma'$ 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 $\{\ell + i \mod n : i = 0, \ldots, d\}$ for some level $\ell$ 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 $\{\ell + i \mod n : i = 0, \ldots, d\}$ for some level $\ell$ and $d \leq km$. □

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

Towards a contradiction, suppose that $G_{F_{km+1}} \models \tilde{Q}$. Then there is a match $H$ of some $q$ from $\tilde{Q}$ in $F_{km+1}$. Since $Q$ is connected, so is $\tilde{Q}$ 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 $G_T \models \tilde{Q}$. But at the same time, $T$ is locally isomorphic to a subframe of $F$, so $T$ weakly refutes $\tilde{Q}$.
refutes $Q$. Because $T$ is a tree, by Lemma 4.1, $T$ actually refutes $Q$. That is, $G_T \not\models \tilde{Q}$. The obtained contradiction shows that $G_{F_{kmz}} \not\models \tilde{Q}$. □

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 $\Theta_f$ of maximal types over $\Gamma_f$ to be respected, a distinguished type $\tau_f \in \Theta_f$ to be realized, a TBox $T_f$ to be satisfied, and a query $Q_f$ to be avoided. We call $(\tau_f, T_f, \Theta_f, Q_f)$ an (abstract) component of $F$.

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

A witnessing graph for component $(\tau_f, T_f, \Theta_f, Q_f)$ is any finite pointed graph $G$ such that $G$ respects $\Theta_f$, the distinguished element of $G$ is of type $\tau_f$, $G \models T_f$, 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 $(\tau_f, T_f, \Theta_f, Q_f)$ with an arbitrary witnessing graph $G_f$, and each edge labelled $(\tau, r)$ from $f$ to $e$ with edges labelled $(v, r)$ from $f$ to $e$, for $v$ ranging over all nodes of type $\tau$ in $G_f$. (Note that connector $G_{f, \tau}$ is then isomorphic to $G_{f', \tau}$ for each node $v$ of type $\tau$ 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 $\tau$ if $\tau \subseteq \tau_f$ for some component $(\tau_f, T_f, \Theta_f, Q_f)$ of $F$. Clearly, if $F$ realizes type $\tau$, 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 $(\tau_f, T_f, \Theta_f, Q_f)$, if a graph does not satisfy $Q_f$, then it refutes $Q$; that is, $Q_f$ contains $\tilde{Q}$ in the query containment sense;

and additionally require that $\Gamma_F$ includes all node labels used in $\tilde{Q}$. 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 $\tau$ (over labels used in $T$ and $\tilde{Q}$) can be realized in a finite graph that satisfies $T$ 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 \( \top \subseteq C_{\rightarrow} \) or \( \top \subseteq C_{\leftarrow} \);
- 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 \( (\tau_f, T_f, \Theta_f, Q_f) \), either each type in \( \Theta_f \) contains \( C_{\rightarrow} \) or each type in \( \Theta_f \) 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 \( A \subseteq \exists r^{-}.B \), and flipping CIs involving universal restrictions over inverse roles: \( A \subseteq \forall r^{-}.B \) is replaced by \( \bar{B} \subseteq \forall r.\bar{A} \). The TBox \( \mathcal{T}_{\rightarrow} \) is defined symmetrically. Note that while \( \mathcal{T}_{\rightarrow} \) 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 \( \{ \top \subseteq C_{\rightarrow} \} \cup \mathcal{T}_{\rightarrow} \) or \( \{ \top \subseteq C_{\leftarrow} \} \cup \mathcal{T}_{\leftarrow} \);
- in each connector \( G_{f,v} \), the distinguished node satisfies \( \{ \top \subseteq C_{\rightarrow} \} \cup \mathcal{T}_{\rightarrow} \) or \( \{ \top \subseteq C_{\leftarrow} \} \cup \mathcal{T}_{\leftarrow} \).

For an alternating abstract frame \( \mathcal{F} \), we replace the first item with

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

and additionally require that \( \Gamma_f \) includes all concept names used in \( \mathcal{T} \). As connectors in alternating frames are directed and CIs are suitably flipped in \( \mathcal{T}_{\rightarrow} \) and \( \mathcal{T}_{\leftarrow} \), the second item implies that each connector satisfies all CIs of the form \( A \subseteq \forall r.B \) or \( A \subseteq \forall r^{-}.B \) 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 \( \mathcal{F} \) satisfies an \( \mathcal{ALCI} \) TBox \( \mathcal{T} \), so does each graph \( \mathcal{F} \) 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 \( \tau \), an \( \text{ALCI} \) TBox \( T \), and a connected UCRPQ \( Q \), type \( \tau \) is realized in a finite graph that satisfies \( T \) and refutes \( Q \) iff \( \tau \) is realized in a productive alternating abstract frame that satisfies \( T \) and weakly refutes \( Q \). [Proof: A.1]

Using Lemma 5.3, we can compute the set of such types \( \tau \) 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 \( \text{ALC} \) \([28]\) (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 \( \text{ALCQ} \) 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 \( \tau \) is realized in a finite graph that satisfies a TBox \( T \) 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 \( \Sigma_T \) be the set of role names used in the input TBox \( T \). Intuitively, we want to reduce an instance of the problem to multiple instances that do not involve \( \Sigma_T \)-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 \( \Sigma_0 \)-reachability if \( G \not\models \hat{Q} \mod \Sigma_0 \) where \( \hat{Q} \mod \Sigma_0 \) is the query obtained from \( Q \) by dropping all \( \Sigma_0 \)-reachability atoms. The problem we will be solving, dubbed finite entailment modulo \( \Sigma_0 \)-reachability, is to decide if a type \( \tau \) can be realized in a finite graph that satisfies \( T \), respects \( \Theta \), and refutes \( Q \) modulo \( \Sigma_0 \)-reachability, where \( \Sigma_0 \supseteq \Sigma_T \). (We recover the original problem by taking \( \Theta = \{\emptyset\} \) and \( \Sigma_0 = \Sigma_T \cup \{r\} \) for some fresh role name \( r \).) The intermediate step will amount to replacing \( \Sigma_0 \)-reachability with \( \Sigma_T \)-reachability. Next, we explain how to factorize \( \text{ALCQ} \) TBoxes over frames, which will be needed in both steps.

Factorizing \( \text{ALCQ} \) TBoxes

Let \( T \) be an \( \text{ALCQ} \) TBox. Let \( \Gamma_T \) 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 \( T \), and each \( n \leq N \) where \( N \) is one plus the maximal number used in \( T \). Let \( T_n \) be the TBox obtained from \( T \) by dropping all CIs involving roles, and adding

\[
\begin{align*}
C_{n,r,D} &\subseteq \exists^n r.D & \text{for each } C_{n,r,D} \in \Gamma_T, \\
\hat{C}_{n,r,D} &\subseteq \exists^n r.D & \text{for each } C_{n,r,D} \in \Gamma_T \text{ with } n < N, \\
\check{C}_{N,r,D} &\subseteq \exists^{n-1} r.D & \text{for each } C_{N,r,D} \in \Gamma_T.
\end{align*}
\]

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

\[
C \subseteq \bigcap_{i=0}^{n} (C_{i,r,D} \cap \exists^{n-i} r.D)
\]
and each $C \subseteq \exists^n r.D$ with
\[
C \subseteq \bigcup_{i=0}^{n} (C_{i,r,D} \cap \exists^{n-i} r.D) \cup \bigcup_{i=n+1}^{N} C_{i,r,D}. 
\]

$\mathcal{T}_e$ 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 $\mathcal{T}_e$;
- in each connector $G_{f,a}$, the distinguished node satisfies $\mathcal{T}_e$ 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 $\mathcal{T}_e$.

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

**From $\Sigma_0$-reachability to $\Sigma_T$-reachability**

Without loss of generality we can assume that countermodels only use role names from $\Sigma_T$. The general idea of this step is to decompose countermodels into strongly connected components. Within each component, all $\Sigma_T$-reachability atoms are trivially satisfied, so if the component refutes $Q$ modulo $\Sigma_0$-reachability, it also refutes $Q$ modulo $\Sigma_T$-reachability (because $\Sigma_T \subseteq \Sigma_0$). Replacing $\Sigma_0$-reachability with $\Sigma_T$-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 $\Sigma_T$-reachability, also refutes $Q$ modulo $\Sigma_0$-reachability (provided $\Sigma_T \subseteq \Sigma_0$). 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 $(\Sigma_1, \Sigma_2)$-reachability if
- each component $G_f$ refutes $Q$ modulo $\Sigma_1$-reachability; that is, $G_f \not\models \hat{Q}$ mod $\Sigma_1$;
- each connector $G_{f,a}$ refutes $Q$ modulo $\Sigma_2$-reachability; that is, $G_{f,a} \not\models \hat{Q}$ mod $\Sigma_2$.

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 $\hat{Q}$ mod $\Sigma_1$ in the query containment sense; and require that $\Gamma_F$ includes all node labels used in $\hat{Q}$.

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

**Lemma 6.3.** Consider a type $\tau$, an $\mathcal{ALCQ}$ TBox $\mathcal{T}$, a set $\Theta$ of types, a connected simple UC2RPQ $Q$, and $\Sigma_0 \supseteq \Sigma_T$. Type $\tau$ is realized in a finite graph that satisfies $\mathcal{T}$, respects $\Theta$, and refutes $Q$ modulo $\Sigma_0$-reachability iff $\tau$ is realized in a productive abstract frame that is a tree, satisfies $\mathcal{T}$, respects $\Theta$, and weakly refutes $Q$ modulo $(\Sigma_T, \Sigma_0)$-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 $\Sigma_T$-reachability, which we handle next.

**Entailment modulo $\Sigma_T$-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 $|\Sigma_T|$, unless it is a $\Sigma_T$-reachability atom.
Let \( r_1, r_2, \ldots, r_m \) be an enumeration of \( \Sigma_T \). For convenience, let \( r_{m+1} = r_1 \). For each \( r \in \Sigma_T \), let \( C_r \) be a fresh concept name. By an \( r \)-node we mean a node that has label \( C_r \) and labels \( \hat{C}_s \) for all \( s \in \Sigma_T \setminus \{r\} \). A concrete frame is role-alternating if

- every component \( G_f \) satisfies
  \[
  \{ T \subseteq C_r \} \cup \{ T \subseteq C_{0,r,D} \mid C_{0,r,D} \in \Gamma_T \} \cup \{ T \subseteq \hat{C}_s \mid s \in \Sigma_T \setminus \{r\} \}
  \]
  for some \( r \in \Sigma_T \);
- every connector \( G_{f,o} \) 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 \( (\tau_f, T_f, \Theta_f, Q_f) \), there is \( r \in \Sigma_T \) such that each \( \tau \in \Theta_f \) contains \( C_r \), all \( C_{0,r,D} \in \Gamma_T \), and all \( \hat{C}_s \) with \( s \in \Sigma_T \setminus \{r\} \).

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 \( \Sigma_T \)-reachability atom, is at most \( |\Sigma_T| \). [Proof: B.4]

**Lemma 6.5.** Consider a type \( \tau \), an \( \mathcal{ALCQI} \) TBox \( T \), a set \( \Theta \) of types, and a connected simple UC2RPQ \( Q \). Type \( \tau \) is realized in a finite graph that satisfies \( T \), respects \( \Theta \), and refutes \( Q \) modulo \( \Sigma_T \)-reachability iff \( \tau \) is realized in a productive role-alternating abstract frame that satisfies \( T \), respects \( \Theta \), and weakly refutes \( Q \) modulo \( (\Sigma_T, \Sigma_T) \)-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 \( \Sigma_T \). Each component of a role-alternating abstract frame that satisfies \( T \), 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 \( \Sigma_T \) with one role fewer. Apart from that, the reduction only affects the type \( \tau \) to realize and the set \( \Theta \) of allowed types, which must account for newly added concept names from \( \Gamma_T \). The size of \( \Gamma_T \) is exponential in the size of the original TBox (not the current TBox \( T \) 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


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 $\tau$, 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 $f'_{u}$ labelled with $G'_{u}^r$ and node $f''_{u}$ 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 $f'_{w}$ to $f''_{v}$ with label $(u_{w}^r, r)$ and from $f''_{w}$ to $f''_{u}$ with label $(v_{w}^r, r)$ for every node $w$ in $G$ and every $s \in \Sigma_{\mathcal{T}}$, where $u_{w}^r$ is the copy of $u$ in $G'_{w}^r$ and $v_{w}^r$ is the copy of $v$ in $G'_{w}^r$. The resulting concrete frame $F$ is alternating by construction and obviously realizes $\tau$. Up to labels $C_{\rightarrow}$ and $C_{\leftarrow}$, 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 $\{\top \subseteq C_{\rightarrow}\} \cup \mathcal{T}_{\leftarrow}$ or $\{\top \subseteq C_{\leftarrow}\} \cup \mathcal{T}_{\rightarrow}$; 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 $\tau$, $\mathcal{T}$, or $\bar{Q}$, plus $C_{\rightarrow}$ (recall that $C_{\rightarrow} = \bar{C}_{\leftarrow}$). We modify $F$ as follows. Consider a frame node $f$ in $F$. For each maximal type $\sigma$ over $\Gamma_{F'}$ realized in $G_{f}$, choose an arbitrary node $v_{\sigma}$ of type $\sigma$ in $G_{f}$. Adjust edges originating in $f$ to ensure that $G_{f,v_{\sigma}}$ is isomorphic to $G_{f,a_{\sigma}}$: edges with labels of the form $(v_{\sigma}, r)$ should be relabelled with $(\sigma, r)$, the remaining ones should be removed. Let $u_{f}$ be the distinguished node in $G_{f}$. Relabel $f$ with $(\tau_{f}, \mathcal{T}_{f}, \Theta_{f}, Q_{f})$ where

- $\tau_{f} = \{A \in \Gamma_{F'}^{+} \mid \sigma_{f} \text{ has label } A \text{ in } G_{f}\}$
- $\mathcal{T}_{f}$ is $\mathcal{T}_{\leftarrow}$ if $\sigma_{f}$ has label $C_{\rightarrow}$ in $G_{f}$, and $\mathcal{T}_{\rightarrow}$ if it has label $C_{\leftarrow}$;
- $\Theta_{f}$ is the set of maximal types over $\Gamma_{F'}$ realized in $G_{f}$;
- $Q_{f} = \bar{Q}$.

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 $\tau$, satisfies $\mathcal{T}$, 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 $\tau$, by Lemma 5.2 it follows that $G_{F'}$ realizes $\tau$, 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 $\tau$, satisfies TBox $\mathcal{T}$, and weakly refutes query $Q$, can be decided in 2EXPSPACE. 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 $C_{\rightarrow} = \bar{C}_{\leftarrow}$, we can restrict our attention to abstract frames involving only types over $\Gamma_{0}$, defined as the set of concept names containing $C_{\rightarrow}$ and all concept names used in $\tau$, $\mathcal{T}$, and $\bar{Q}$. Consider the set $\Psi_{\mathcal{T},Q}$ of all maximal types over $\Gamma_{0}$ 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 $\Gamma_{0}$.
that are realized in a component of such a frame, not necessarily in the distinguished node. Indeed, for a type \( \sigma \) 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 \( e \), copying all the outgoing edges from \( f \), and the result will still be an alternating concrete frame that satisfies \( T \) and weakly refutes \( Q \).

For a set \( \Psi \) of maximal types over \( \Gamma_0 \), let \( \Psi_\rightarrow \) and \( \Psi_\leftarrow \) denote the sets of types from \( \Psi \) that contain \( C_\rightarrow \) and \( C_\leftarrow \), respectively. The set \( \Psi_{\rightarrow,\leftarrow} \) of maximal types over \( \Gamma_0 \) such that for each \( \sigma \in \Psi \):

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

Note that, for a fixed \( \sigma \), the conditions are monotone w.r.t. \( \Psi \): that is, if they hold for \( \Psi \), they also hold for each \( \Psi' \supseteq \Psi \). Consequently, we can compute \( \Psi_{\rightarrow,\leftarrow} \) by a simple greatest fixed point procedure. For \( i = 0, 1, 2, \ldots \), we iteratively compute sets \( \Psi_i \) of maximal types \( \sigma \) over \( \Gamma_0 \) that satisfy the conditions above with \( \Psi = \Psi_{i-1} \), starting from the set \( \Psi_0 \) of all maximal types over \( \Gamma_0 \). We stop when \( \Psi_{i+1} = \Psi_i \) and let \( \Psi_{\rightarrow,\leftarrow} = \Psi_i \). Then, it suffices to check whether \( \Psi_{\rightarrow,\leftarrow} \) contains a type \( \sigma \) such that \( \sigma \supseteq \tau \).

For a given \( \sigma \) 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 \( \mathcal{ALL} \) 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 \( \tilde{Q} \), 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 \( \sigma \) 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 \( T \). 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 \( \tilde{Q} \) over the connector, which can be done in time \( O(|\tilde{Q}| \cdot m^n \cdot n \cdot m \cdot k) \), where \( n \) is the maximal size of a CRPQ in \( \tilde{Q} \) (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 \( \tau \), a TBox \( T \) mentioning no roles, a set \( \Theta \) of types, a connected simple UC2RPQ \( Q \), and \( \Sigma_1 \subseteq \Sigma \), decide if \( \tau \) is realized in a graph that satisfies \( T \), has only nodes of types from \( \Theta \), and does not satisfy \( \tilde{Q} \mod \Sigma_0 \). Because \( T \) 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 \( \tau, \mathcal{T}, \Theta, \) 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 \( \Sigma_0 \) reachability.

**Lemma B.1.** If a concrete frame \( F \) is a tree and weakly refutes a connected simple UC2RPQ \( Q \) modulo \((\Sigma_0, \Sigma_0)\)-reachability, then \( G_F \) refutes \( Q \) modulo \( \Sigma_0 \)-reachability.

**Proof.** Just like for \( \widehat{Q} \), we can prove that \( \widehat{Q} \) mod \( \Sigma_0 \) is factorized. Given that, the claim follows immediately from Lemma 4.2 for \( P = \widehat{Q} \) mod \( \Sigma_0 \).

**Proof of Lemma 6.3.** For the left-to-right implication, consider a graph \( G \) that realizes \( \tau \), satisfies \( \mathcal{T} \), has only nodes of types from \( \Theta \), and refutes \( Q \) modulo \( \Sigma_0 \)-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 \( f_0 \) labelled with \( G_v \) for each node \( v \) in \( G \). Whenever there is an \( r \)-edge from \( u \) to \( v \), add an edge from \( f_v \) to \( f_u \) 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 \( \tau \) (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 \( \tau \) and all its components only contain nodes of types in \( \Theta \). 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 \( \Theta \).

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 \( \Sigma_0 \)-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 right-to-left 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 \( \tau, \mathcal{T}, \Theta, \) and \( \widehat{Q} \), plus all node labels of the form \( C_{n,r,D} \) used in \( \mathcal{T}_n \) and \( \mathcal{T}_r \). For each frame node \( f \), we let \( T_f = T_\mathcal{T} \) 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 \( \tau \) and contains only nodes of types from \( \Theta \). Because \( F \) refutes \( Q \) modulo \( (\Sigma_\mathcal{T}, \Sigma_0) \)-reachability and \( \Sigma_\mathcal{T} \subseteq \Sigma_0 \), \( F \) also refutes \( Q \) modulo \( (\Sigma_0, \Sigma_0) \)-reachability. By Lemma B.1, \( G_F \) refutes \( Q \) modulo \( \Sigma_0 \)-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 $\Psi_1$ of types appearing in leaves of such abstract frames; the set $\Psi_2$ of types appearing in leaves or their parents, etc.

Let $\Gamma_0$ be the set of concept names in $\tau$, $\mathcal{T}$, $\Theta$, and $C_T$. Let $\Psi_0 = \emptyset$. For $i > 0$, $\Psi_i'$ is defined as the set of maximal types $\sigma$ over $\Gamma_0$ such that there $\sigma$ contains a type from $\Theta$ and there is a connector with the distinguished element of type $\sigma$, non-distinguished elements of types from $\Psi_{i-1}$, 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 $\Psi_i$ is defined as the set of types $\sigma$ such that the abstract component $(\sigma, \mathcal{T}_i, \Psi_i', \widehat{Q} \mod \Sigma_T)$ is productive. That is, we must ensure that $\sigma$ is realizable in a finite graph that satisfies $\mathcal{T}_i$, respects $\Psi_i'$, and refutes $Q$ modulo $\Sigma_T$-reachability atoms, which is precisely the problem we want to reduce to. We compute $\Psi_i$ iteratively for $i = 1, 2, \ldots$ until $\Psi_i = \Psi$; the resulting set $\Psi$ contains a type $\sigma$ 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 every 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 $\Sigma_T$-reachability atom, there exist role names $r, s \in \Sigma_T$ such that $r \notin R$ and $s \notin R$ (since otherwise $\Sigma_T \subseteq R$ or $\Sigma_T \subseteq R$). Let $\sigma$ be an undirected path in a role-alternating frame $F$ induced by a path witnessing such a 2RPQ in $\tilde{G}_F$. Consider the sequence of roles in the labels of edges traversed by $\sigma$, 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 $(r_k, r_{k+1})$, $(r_{k+1}, r_k)$, $(r_k, r^-_k)$, $(r^-_k, r_k)$ or $(r^-_k, r_{k-1})$ 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 $\sigma$. Since the role name $r$ and inverse role $s^-$ are not mentioned by the 2RPQ, the length of the sequence is bounded by $|\Sigma_T| - 1$. We can apply the same reasoning to every infix of $\sigma$, proving that the span of $\sigma$ is at most $|\Sigma_T| - 1$.

### B.5 Proof of Lemma 6.5

Before proving Lemma 6.5, we need to generalize Lemma 4.3 to refuting modulo $\Sigma_0$-reachability.

**Lemma B.2.** Let $F$ be a concrete frame all of whose nodes are reachable from a single node $\dagger$ and let $Q$ be a connected simple UC2RPQ. If $F$ weakly refutes $Q$ modulo $(\Sigma_0, \Sigma_0)$-reachability and all 2RPQs in $\widehat{Q}$ that are not $\Sigma_0$-reachability atoms have bounded span in $F$, then there is a frame locally isomorphic to $F$ that actually refutes $Q$ modulo $\Sigma_0$-reachability.

**Proof.** There exists $k > 0$ such that, for each 2RPQ $p$ in $\widehat{Q}$ that is not a $\Sigma_0$-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 $\text{Coil}(F, km + d)$ 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 $\text{Coil}(F, km + d)$ isomorphic with $\text{Unravel}(F, km + d - 1, f_0)$. 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 $\text{Unravel}(F, km, f)$ for each node $f$ in $F$.

\dagger 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 $\tilde{Q}$ mod $\Sigma_0$ in $\text{Coil}(F, km + d)$ is a match of some $C2RPQ$ $q \in \tilde{Q}$ mod $\Sigma_0$, 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 $\text{Unravel}(F, km, v)$ for some node $v$ in $F$. Thus, every match of $\tilde{Q}$ mod $\Sigma_0$ in $\text{Coil}(F, km + d)$ maps homomorphically to $T$. By converting $T$ to a frame we obtain a tree-shaped frame that weakly refutes $Q$ modulo $(\Sigma_0, \Sigma_0)$-reachability; by Lemma B.1, $\hat{G}_T$ refutes $Q$ modulo $\Sigma_0$-reachability, which proves that there is no match of $\tilde{Q}$ mod $\Sigma_0$ in $\text{Coil}(F, km + d)$.

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

Suppose first that $|\Sigma_T| > 1$. Let $r_1, r_2, \ldots, r_m$ be an enumeration of $\Sigma_T$ 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 $G_{0,r_i}$ 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 $f_{0,r_i}$ labelled with $G_{0,r_i}$ 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 $f_{u,r_i}$ to $f_{v,r_i}$, 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 $T$. Frame $F$ obviously realizes $\tau$ and all its components have only nodes of types from $\Theta$ (node labels $C_{n,r,D}$ are fresh, so they do not matter for $\Theta$). 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 $\Sigma_T$-reachability.

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

Suppose now that $\Sigma_T = \{r\}$. This time, for each $u$ in $G$, and each $i \in \{0, 1\}$, we let $G^i_u$ 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 graph with nodes $f^0_u$ and $f^1_u$ for each node $v$ in $G$, labelled with $G^0_u$ and $G^1_u$, respectively. Whenever there is an $r$-edge from $u$ to $v$ in $G$, we add an edge from $f^0_u$ to $f^1_u$ labelled with $(u^0, r)$ and an edge from $f^1_u$ to $f^0_u$ 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 $|\Sigma_T| > 1$.

For the right-to-left implication, take such a productive abstract frame and consider a concrete frame $F$ it represents. Frame $F$ realizes $\tau$, satisfies $T$, weakly refutes $Q$ modulo $\Sigma_T$-reachability, and its components only contain nodes of types from $\Theta$. 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 $\tau$. 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 $\Gamma_0$ be the set containing all concept names used in $\tau$, $T$, $\Theta$, and $\tilde{Q}$, as well as those from $\Gamma_T$ and all $C_r$ with $r \in \Sigma_T$. 

We aim to compute the set \( \Psi_{T,\Theta,\mathcal{Q}} \) of maximal types over \( \Gamma_0 \) that are realized in abstract frames with properties specified in Lemma 6.5. For a set \( \Psi \) of types over \( \Gamma_0 \) and a role name \( r \in \Sigma_T \), let 
\[ \Psi_r = \{ \sigma \in \Psi \mid C_r \in \sigma \} \].
Let \( r_1, r_2, \ldots, r_m \) be an enumeration of \( \Sigma_T \). The set \( \Psi_{T,\Theta,\mathcal{Q}} \) is the greatest set \( \Psi \) of maximal types over \( \Gamma_0 \) such that 
\[ \Psi = \bigcup_{i=1}^{m} \Psi_{r_i} \]
and for each \( i \in \{1, 2, \ldots, m\} \) and \( \sigma \in \Psi_{r_i} 
\begin{itemize}
  \item \( \sigma \) contains all concept names \( C_{0,r_i,D} \in \Gamma_T \) and \( \bar{C}_{r_j} \) with \( j \neq i \);
  \item \( \sigma \supseteq \sigma' \) for some \( \sigma' \in \Theta \);
  \item abstract component \( (\sigma, \mathcal{T}_n, \Psi_{r_i}, \hat{\mathcal{Q}} \mod \Sigma_T) \) is productive;
  \item there is a connector with distinguished element of type \( \sigma \) and non-distinguished elements of types from \( \Psi_{r_{1+i}} \), that satisfies all properties required for connectors of witnessing frames in Lemma 6.5.
\end{itemize}
We can compute it using a greatest fixed-point procedure, as in Section A.2, starting from the set of all maximal types over \( \Gamma_0 \) that, for some \( i \in \{1, 2, \ldots, m\} \), contain \( C_{r_i} \), all \( C_{0,r_i,D} \in \Gamma_T \), 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 \( (\sigma, \mathcal{T}_n, \Psi_{r_i}, \hat{\mathcal{Q}} \mod \Sigma_T) \). Because it is always guaranteed that all types in \( \Psi_{r_i} \) contain all \( C_{0,r_i,D} \in \Gamma_T \), we can drop from \( \mathcal{T}_n \) all CIs that involve role \( r_i \). This way we reduce the productivity test to finite entailment modulo \( \Sigma_T \), 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 \( \mathcal{T}_n \), which has exponential size and introduces exponentially many fresh concept names \( \Gamma_T \). Deeper in the recursion, we re-apply the same construction to \( \mathcal{T}_n \). 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 \( (\mathcal{T}_n)_n \), 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 \( \tau \) to realize and set \( \Theta \) 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