Revisiting Conjunctive Query Entailment for $\mathcal{S}$

Yazmín Ibáñez-García[1], Jean Christoph Jung[2], Vincent Michielini[3], Filip Murlak[4]

[1]Cardiff University
[2]TU Dortmund University
[3]University of Bordeaux
[4]University of Warsaw
ibanezgarciay@cardiff.ac.uk, jean.jung@tu-dortmund.de, vincent.michielini@u-bordeaux.fr, f.murlak@uw.edu.pl

Abstract

We clarify the complexity of answering unions of conjunctive queries over knowledge bases formulated in the description logic $\mathcal{S}$, the extension of $\mathcal{ALC}$ with transitive roles. Contrary to what existing partial results suggested, we show that the problem is in fact 2ExpTime-complete; hardness already holds in the presence of two transitive roles and for Boolean conjunctive queries. We complement this result by showing that the problem remains in coNExpTime when the input query is rooted or is restricted to use at most one transitive role (but may use arbitrarily many non-transitive roles).

1 Introduction

In this paper, we aim to complete the complexity landscape for the problem of query answering over knowledge bases expressed in expressive description logics (DLs), that is, logics extending the basic DL $\mathcal{ALC}$. As is common in such endeavor, we focus on the associated decision problem known as *query entailment*: given a DL knowledge base (KB) consisting of an ABox and a TBox, a query, and a tuple of individuals, the goal is to determine whether the query returns this tuple in every model of the given KB. As query languages we consider *unions of conjunctive queries (UCQs)* and fragments thereof. This problem has been heavily studied and is well understood. Existing results suggest a dichotomy:

- The problem is ExpTime-complete for $\mathcal{ALC}$ and its extensions with role hierarchies ($\mathcal{H}$) or qualified number restrictions ($\mathcal{Q}$) (Lutz 2008; Ortiz, Simkus, and Eiter 2008).
- The problem is 2ExpTime-complete for the extension $\mathcal{ALC}_{self}$ of $\mathcal{ALC}$ with the *self*-constructor (Bednarczyk and Rudolph 2023), and for every extension that allows either inverse roles ($\mathcal{I}$) or both $\mathcal{H}$ and transitive roles ($\mathcal{S}$), as long as it remains inside the DL $\mathcal{SHIQ}$, which combines the extensions $\mathcal{S}, \mathcal{H}, \mathcal{I}, \mathcal{Q}$ (Eiter et al. 2009; Glimm, Horrocks, and Sattler 2008).

Often, the restriction to *rooted* queries, that is, connected queries with answer variables, leads to lower complexity. This is the case for $\mathcal{ALC}_{self}$ (Bednarczyk 2024) and all DLs between $\mathcal{ALCI}$ and $\mathcal{SHIQ}$, where rooted query entailment is coNExpTime-complete as long as transitive role names are disallowed in queries (Lutz 2008). Without inverses and *self*, the complexity is even lower: entailment of (rooted or not) queries without transitive roles is ExpTime-complete for all DLs between $\mathcal{ALC}$ and $\mathcal{SHQ}$ (Lutz 2008). Working with rooted queries does not necessarily help if transitive roles are allowed in queries: analyzing the 2ExpTime-hardness proof for query entailment in $\mathcal{SH}$ from (Eiter et al. 2009) shows that rooted query entailment remains 2ExpTime-complete.

A notable gap remains for $\mathcal{S}$ without any extensions. For $\mathcal{S}$ the problem has been proven to be coNExpTime-complete when KBs and queries are allowed to use a single transitive role (and no other roles). The lower bound comes from (Eiter et al. 2009) and the upper one from (Bienvenu et al. 2010). A 2ExpTime-upper bound for the general case follows from the mentioned result for $\mathcal{SH}$ (Eiter et al. 2009). In the same paper, Eiter et al. also claim an ExpTime upper bound for the case where the ABox is tree-shaped. These results have been regarded as a strong indication that the whole problem may be coNExpTime-complete, challenging the apparent dichotomy (Bienvenu et al. 2010). The complexity for rooted query entailment was open until now, but a coNExpTime-lower bound follows from (Eiter et al. 2009).

The aim of this paper is to revisit the query entailment problem for $\mathcal{S}$ and close the mentioned gaps. Our first main result is that, surprisingly, when at least two transitive roles are allowed in the query, the problem is 2ExpTime-hard already for $\mathcal{S}$ (without role hierarchies), and hence 2ExpTime-complete. Importantly, our lower bound works with a tree-shaped ABox and thus contradicts the mentioned ExpTime upper bound for that case (Eiter et al. 2009). Indeed, in their argument, Eiter et al. compile input conjunctive queries (CQs) to so-called *pseudo-tree queries (PTQs)*, designed to capture the behavior of CQs over tree-like interpretations, which is sufficient due to the tree-like model property of $\mathcal{S}$. The error seems to arise from a subtle mismatch between the formal definition of PTQs used in the compilation process, and their intuitive understanding as *trees of clusters*, on which the later algorithmic treatment relies.

Besides the presence of two transitive roles in the query, our 2ExpTime-hardness proof relies on the availability of non-rooted queries. We complement our lower bound by showing that both conditions are necessary: UCQ entailment remains in coNExpTime if at most one transitive role is allowed in the query or if the query is rooted.

We develop the two proofs in parallel, using common terminology and data structures whenever possible. At the core, we show a *small witness property* in the following sense: the query is not entailed iff there is a small structure witnessing that. Note that this structure cannot be simply a countermodel since UCQ entailment for $\mathcal{S}$ is not finitely controllable (Rosati 2011). The NEXPTIME-algorithm for *non*-entailment can then just guess a small structure and verify that it is indeed a witness. Our argument is divided into three steps.

Step 1. Reduce UCQ entailment to a special case with trivial ABoxes and UCQs of special shape.

Step 2. Reduce the special case of entailment to the existence of structures called *mosaics*, which are collections of interpretations called *tiles* that respect certain compatibility requirements.

Step 3. Show that both the size of all tiles in a mosaic and their number can be bounded exponentially.

While the overall strategy is familiar, the steps are subtle. Steps 1–2 for the single transitive role case rely on a careful refinement of Eiter et al.'s PTQs, which ensures correctness of the algorithmic treatment, but makes the compilation process significantly harder. The crux of Step 3 is to show that UCQ entailment *is* finitely controllable as long as the set of used role names consists of a single transitive role name and queries are acyclic. Going beyond existing results (Bienvenu et al. 2010), we show that the size of the countermodel can be bounded independently from the query.

The structure of the paper is as follows. We give the necessary preliminaries in Section 2. Section 3 contains the proof of the 2EXPTIME-lower bound. Section 4 focuses on the case of a single transitive role and acyclic queries, needed for Step 3. In Section 5 we revise the notion of PTQs. In Section 6 we implement the three steps and obtain both upper bounds. We conclude in Section 7.

Proofs for all statements in the paper are available in the full version (Ibáñez-García et al. 2025).

2 Preliminaries

TBoxes, ABoxes, and Knowledge Bases. We fix countably infinite sets $\mathsf{N_I}$ of *individual names*, $\mathsf{N_C}$ of *concept names*, and $\mathsf{N_R}$ of *role names*, partitioned into non-transitive role names $\mathsf{N_R^{nt}}$ and transitive role names $\mathsf{N_R^t}$. *Concepts* C of the description logic $\mathcal{S}$ are defined by the grammar:

$$C ::= A \mid \neg C \mid C_1 \sqcup C_2 \mid \exists r.\, C\,.$$

A *concept inclusion (CI)* is an expression of the form $C \sqsubseteq D$ for concepts C, D. A *TBox* is a finite set of concept inclusions. An *ABox* is a finite set of concept assertions $A(a)$ and role assertions $r(a,b)$ for $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $a, b \in \mathsf{N_I}$. A *knowledge base (KB)* is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We write $\mathsf{N_C}(\mathcal{T})$, $\mathsf{N_I}(\mathcal{A})$, etc. for the finite sets of concept names, individual names, etc. that occur in a particular TBox $\mathcal{T}$ or ABox $\mathcal{A}$.

Interpretations. The semantics of concepts, TBoxes, and ABoxes are defined as usual based on *interpretations*. An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is the *domain* of $\mathcal{I}$; and $\cdot^{\mathcal{I}}$ assigns a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of the domain to every concept name $A \in \mathsf{N_C}$, a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every role name $r \in \mathsf{N_R}$, and an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to every individual name $a \in \mathsf{N_I}$ (Baader et al. 2017). The interpretation of complex concepts is standard:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},$$
$$(\exists r.\, C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \langle d, e \rangle \in r^{\mathcal{I}} \text{ for some } e \in C^{\mathcal{I}}\}.$$

An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $t^{\mathcal{I}}$ is a transitive relation for all $t \in \mathsf{N_R^t}$, and $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every concept inclusion $C \sqsubseteq D$ in $\mathcal{T}$. For ABoxes, we adopt the *standard name assumption*; that is, we assume $a^{\mathcal{I}} = a$ for all $a \in \mathsf{N_I}(\mathcal{A})$. Then, $\mathcal{I}$ is a *model* of ABox $\mathcal{A}$, written $\mathcal{I} \models \mathcal{A}$, if $a \in A^{\mathcal{I}}$ for every assertion $A(a) \in \mathcal{A}$ and $\langle a, b \rangle \in r^{\mathcal{I}}$ for every assertion $r(a,b) \in \mathcal{A}$. Finally, $\mathcal{I}$ is a model of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.

For a domain element $d \in \Delta^{\mathcal{I}}$, the *type of d in $\mathcal{I}$* is defined as $\mathrm{tp}(\mathcal{I}, d) = \{A \in \mathsf{N_C} \mid d \in A^{\mathcal{I}}\}$.

The *transitive closure* of an interpretation $\mathcal{I}$ is the interpretation $\mathcal{I}^+$ that coincides with $\mathcal{I}$ except that, for every $t \in \mathsf{N_R^t}$, $t^{\mathcal{I}^+}$ is the transitive closure of $t^{\mathcal{I}}$.

A *tree* is a directed acyclic graph in which exactly one node (the *root*) has no incoming edges, and each other node has exactly one incoming edge (originating in the *parent* of the node). With an interpretation $\mathcal{I}$ we associate a directed multigraph $G_{\mathcal{I}}$ in which nodes are the domain elements and edges are obtained by taking the disjoint union of the interpretations of all role names. We call $\mathcal{I}$ *tree-shaped* if $G_{\mathcal{I}}$ is a tree (in particular, it has no parallel edges). We call $\mathcal{I}$ a *transitive-tree* interpretation if it is the transitive closure of a tree-shaped interpretation $\mathcal{I}_0$. The root of $\mathcal{I}$ is the root of $\mathcal{I}_0$.

Conjunctive Queries. Let $\mathsf{N_V}$ be a countably infinite set of *variables*. A *conjunctive query (CQ)* is an expression of the form $q(\bar{x})$ where q is a finite set of *atoms* of the form $A(x)$ or $r(x,y)$ where $x, y \in \mathsf{N_V}$, $A \in \mathsf{N_C}$, and $r \in \mathsf{N_R}$, and $\bar{x}$ is a tuple of variables occurring in the atoms of q. We call $\bar{x}$ the *answer variables* of $q(\bar{x})$. We write $\mathrm{var}(q)$ for the set of all variables occurring in q. A *union of conjunctive queries (UCQ)* $Q(\bar{x})$ is a finite set of CQs with the same answer variables $\bar{x}$, which we call the answer variables of Q. A (U)CQ is *Boolean* if its tuple of answer variables is empty, and *unary* if it is a singleton. We identify a Boolean CQ q with its set of atoms.

A *match* of a CQ $q(\bar{x})$ in an interpretation $\mathcal{I}$ is a function $\delta \colon \mathrm{var}(q) \to \Delta^{\mathcal{I}}$ such that $\delta(x) \in A^{\mathcal{I}}$ for each $A(x) \in q$, and $\langle \delta(x), \delta(y) \rangle \in r^{\mathcal{I}}$ for each $r(x,y) \in q$. For a tuple $\bar{d}$ of domain elements from $\Delta^{\mathcal{I}}$, we write $\langle \mathcal{I}, \bar{d} \rangle \models q(\bar{x})$ if there is a match δ of q in $\mathcal{I}$ with $\delta(\bar{x}) = \bar{d}$. For UCQs, we write $\langle \mathcal{I}, \bar{d} \rangle \models Q(\bar{x})$ if $\langle \mathcal{I}, \bar{d} \rangle \models q(\bar{x})$ for some $q(\bar{x}) \in Q(\bar{x})$. In the Boolean case, we write $\mathcal{I} \models q$ if there is a match of q in $\mathcal{I}$, and $\mathcal{I} \models Q$ if $\mathcal{I} \models q$ for some $q \in Q$.

To any CQ q one can associate a directed multigraph G_q, where nodes represent variables and edges are formed by binary atoms. If atoms share the same pair of variables, this creates parallel edges in G_q. A query $q(\bar{x})$ is called *acyclic* or *connected* if G_q is acyclic or connected, respectively. A query is said to be *rooted* if it is connected and not Boolean. These definitions extend to UCQs: a UCQ Q is *acyclic*, *connected*,

or *rooted* if each CQ in Q is. We call *tree query* (TQ) any unary CQ $q(x)$ such that G_q is a directed tree (in particular, it has no parallel edges), x being its root. A *union of tree queries* (UTQ) is a UCQ that contains only TQs.

For a CQ $q(\bar{x})$, a variable $z \in \mathrm{var}(q)$ is *initial in* q if z has no incoming edges in G_q. For instance, if $q(x)$ is a TQ, then x is initial in $q(x)$.

Query Entailment. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a KB, $Q(\bar{x})$ be a UCQ and $\bar{a}$ be a tuple of individuals from $\mathsf{N_I}(\mathcal{A})$. We say that $\mathcal{K}$ *entails* $Q(\bar{a})$, written $\mathcal{K} \models Q(\bar{a})$, if $\langle \mathcal{I}, \bar{a} \rangle \vDash Q$ for every model $\mathcal{I}$ of $\mathcal{K}$. We study the reasoning problems of UCQ entailment and rooted UCQ entailment. *UCQ entailment* asks, given a KB $\mathcal{K}$ and a Boolean UCQ Q, whether $\mathcal{K} \models Q$; and *rooted UCQ entailment* asks, given a KB $\mathcal{K}$, a rooted UCQ $Q(\bar{x})$, and a tuple $\bar{a}$ from $\mathsf{N_I}(\mathcal{A})$, whether $\mathcal{K} \models Q(\bar{a})$. We also consider the variant *over a single transitive role* t which means that the only role that occurs in $\mathcal{T}$ and Q is t. More general query answering problems, for example, for UCQs with constants, can be reduced to the above using standard methods (Glimm et al. 2008).

Throughout the paper, we assume that TBoxes $\mathcal{T}$ be in *normal form* which means that each concept inclusion in $\mathcal{T}$ has one of the following shapes:

$$\textstyle\bigsqcap_i A_i \sqsubseteq \bigsqcup_j B_j, \quad A \sqsubseteq \exists r.B, \quad A \sqsubseteq \forall r.B,$$

where $A, A_i \in \mathsf{N_C} \cup \{\top\}$, $B \in \mathsf{N_C}$, $B_j \in \mathsf{N_C} \cup \{\bot\}$; $\bot$, $\top$, $\sqcap$, and $\forall$ are part of the syntax, with standard semantics. It is routine to show that this is without loss of generality for the considered entailment problems.

When stating complexity bounds, we write $\|\mathcal{T}\|$ and $\|Q\|$ for the size of $\mathcal{T}$ and Q, respectively, represented as a word over a suitable alphabet.

3 Two Transitive Roles

In this section, we show that UCQ entailment in $\mathcal{S}$ is 2ExpTime complete when the query involves at least two transitive roles. The 2ExpTime-upper bound follows from several previous works, e.g., (Glimm et al. 2008; Calvanese, Eiter, and Ortiz 2014; Gutiérrez-Basulto et al. 2023; Gottlob, Pieris, and Tendera 2013). We focus on the hardness of the problem.

Theorem 1. *UCQ Entailment in $\mathcal{S}$ is* 2ExpTime*-complete. It is* 2ExpTime*-hard already for CQs and ABoxes of the form $\{A(a)\}$, if at least two transitive roles are available.*

Our proof closely follows the 2ExpTime-hardness proof for CQ entailment in $\mathcal{SH}$ provided in (Eiter et al. 2009). Since a simple reduction from this problem seems impossible, we give a direct argument. Here we sketch 2ExpTime-hardness for *unions* of CQs, which is significantly easier than for CQs.

We reduce from the word problem for exponential-space alternating Turing machines. We encode runs of such machines as interpretations of the form shown in Figure 1, where edges labeled by α represent paths of length 2, consisting of a t_1-edge followed by a t_2-edge for $t_1, t_2 \in \mathsf{N_R^t}$. Each gray triangle represents a configuration: existential ones have one successor, and universal two. Each of these triangles is a full binary tree of height n, built from α-edges, whose 2^n leaves correspond to tape cells. A tape cell is encoded using the blue

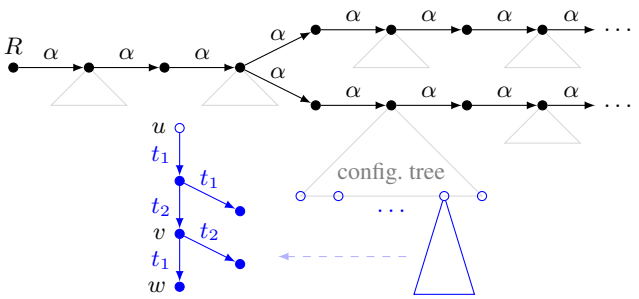

Figure 1: Encoding runs of alternating Turing machines.

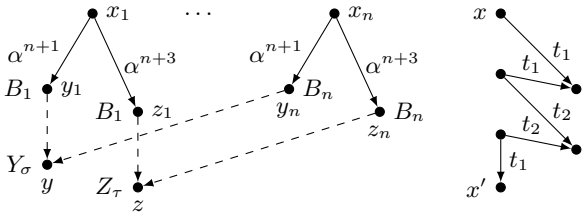

Figure 2: Detecting copying errors.

gadget on the left. The content of the cell is stored in node w: the current using concept names Y_σ and the previous one with Z_σ where σ ranges over possible cell contents. Nodes u and v encode the number of the cell in binary using concept names $B_1, \ldots, B_n$. Each B_i is present in exactly one of the nodes u and v: in u if the ith bit is 0, and in v if it is 1.

With a bit of effort one can define a TBox (using additional concept names to propagate information) that ensures that the interpretation correctly encodes a run of the machine, provided that previous tape content (along with the state annotation) is correctly copied from the previous configuration. The latter is ensured by the query, which is the union of CQs $q_{\sigma,\tau}$ for σ, τ ranging over pairs of *different* cell contents. Each $q_{\sigma,\tau}$ detects a copying error where σ was replaced by τ. It is shown in Figure 2, with the dashed edges representing the path on the right, directed from x to x'.

Variables y and z can only match in consecutive configurations, in the w nodes of some cell gadgets, say w_σ and w_τ. Then, y_i can match only in u_σ or v_σ, and z_i only in u_τ or v_τ. Moreover, owing to the rigidity of the paths α^{n+1} and α^{n+3}, y_i matches in u_σ iff z_i matches in u_τ. Hence, y and z can only match in the same cell of two consecutive configurations. They do if and only if this cell was copied incorrectly.

4 Acyclic Queries, Single Transitive Role

We now focus on the special case of *acyclic* queries over a single transitive role, which is the crux of the problem. We prove an exponential countermodel property.

Theorem 2. *Consider an acyclic Boolean UCQ Q and a TBox $\mathcal{T}$, both of which use a single role name t, which is transitive. Then, for every transitive-tree interpretation $\mathcal{I}$ over t with root d_0 such that both $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models Q$, there exists an interpretation $\mathcal{J}$ satisfying the following:*

- $\Delta^{\mathcal{J}} \subseteq \Delta^{\mathcal{I}}$ *and* $\mathrm{tp}(\mathcal{J}, d) = \mathrm{tp}(\mathcal{I}, d)$ *for each* $d \in \Delta^{\mathcal{J}}$*;*
- $\mathcal{J}$ *is finite and* $|\Delta^{\mathcal{J}}| \leq (|\mathsf{N_C}(\mathcal{T})| + 1)!$*;*
- $d_0 \in \Delta^{\mathcal{J}}$*,* $\mathcal{J} \models \mathcal{T}$*, and* $\mathcal{J} \not\models Q$*.*

Unexpectedly, yet crucially, the bound does not depend on the query at all, unlike in (Bienvenu et al. 2010). Moreover, it depends only on the number of concept names mentioned in $\mathcal{T}$, rather than all concept names occurring in $\mathcal{I}$.

The rest of the section is devoted to the proof of Theorem 2. Let us fix Q, $\mathcal{T}$, $\mathcal{I}$, and d_0 as in the statement. While $\mathcal{I}$ may well be infinite, the extracted interpretation $\mathcal{J}$ will be finite, but not necessarily a transitive-tree interpretation. For $d \in \Delta^{\mathcal{I}}$, define $t^{\mathcal{I}}(d) = \left\{e \in \Delta^{\mathcal{I}} \mid \langle d, e\rangle \in t^{\mathcal{I}}\right\}$ and let $\mathcal{I}_d$ be the subinterpretation of $\mathcal{I}$ induced by $\{d\} \cup t^{\mathcal{I}}(d)$.

Let $m = |Q|$. We define a function $\bar{\mathbf{q}}$ that assigns to each $d \in \Delta^{\mathcal{I}}$ an m-tuple $\bar{q}_d = \langle q_{d,1}, \ldots, q_{d,m}\rangle$ of acyclic Boolean CQs that are *forbidden* in $\mathcal{I}_d$, in the sense that

$$\mathcal{I}_d \not\models \{q_{d,i} \mid 1 \leq i \leq m\} \text{ for each } d \in \Delta^{\mathcal{I}}. \tag{1}$$

The function $\bar{\mathbf{q}}$ is defined by induction. First, we set $\bar{q}_{d_0}$ as any m-tuple listing all CQs from Q. We then proceed top-down, maintaining the invariant. Suppose that $\bar{q}_d$ is already defined for some $d \in \Delta^{\mathcal{I}}$; we shall define $\bar{q}_e$ for all *direct* t*-successors* e of d in $\mathcal{I}$: elements $e \in t^{\mathcal{I}}(d)$ such that there is no $f \in t^{\mathcal{I}}(d)$ with $e \in t^{\mathcal{I}}(f)$. For each $i \leq m$, let $X_{d,i} \subseteq \mathrm{var}(q_{d,i})$ be the set of initial variables of $q_{d,i}$ that can be matched in d; that is, it contains an initial variable x of $q_{d,i}$ iff $d \in A^{\mathcal{I}}$ for each atom $A(x)$ in $q_{d,i}$. Consider the query q' obtained from $q_{d,i}$ by dropping all atoms involving a variable from $X_{d,i}$, and let us look at its connected components. If each connected component of q' admitted a match in a (strict) subtree of $\mathcal{I}_d$ then by merging these matches and mapping each variable from $X_{d,i}$ to d, we would get a match of $q_{d,i}$ in $\mathcal{I}_d$, contradicting the assumption. Hence, there must be a connected component $q'_{d,i}$ of q' that does not admit such a match. (Notice that $q'_{d,i} = q_{d,i}$ iff $X_{d,i} = \emptyset$.) We let $q_{e,i} = q'_{d,i}$ for each direct t-successor e of d in $\mathcal{I}$. Crucially, $\bar{\mathbf{q}}$ is anti-monotone: $\bar{q}_e \preceq \bar{q}_d$ for each $\langle d, e\rangle \in t^{\mathcal{I}}$, where $\bar{q}_e \preceq \bar{q}_d$ iff $q_{e,i}$ is a subquery of $q_{d,i}$ for each i.

For our construction of $\mathcal{J}$, we also formalize the notion of 'visible concepts' of an element in the tree. For each $d \in \Delta^{\mathcal{I}}$, we define $\mathsf{VC}^{\mathcal{I}}_t(d)$ as the set

$$\{A \in \mathsf{N_C}(\mathcal{T}) \mid e \in A^{\mathcal{I}} \text{ for some } e \in t^{\mathcal{I}}(d)\}.$$

Because of the transitivity of $t^{\mathcal{I}}$, the function $\mathsf{VC}^{\mathcal{I}}_t$ is also anti-monotone: $\mathsf{VC}^{\mathcal{I}}_t(e) \subseteq \mathsf{VC}^{\mathcal{I}}_t(d)$ for every $\langle d, e\rangle \in t^{\mathcal{I}}$.

Now that we defined the functions $\bar{\mathbf{q}}$ and $\mathsf{VC}^{\mathcal{I}}_t$, we can finally use them to construct, for each $d \in \Delta^{\mathcal{I}}$, a finite interpretation $\mathcal{J}_d$ with the following properties:

- $d \in \Delta^{\mathcal{J}_d} \subseteq \Delta^{\mathcal{I}}$,
- $\mathrm{tp}(\mathcal{J}_d, e) = \mathrm{tp}(\mathcal{I}, e)$ for all $e \in \Delta^{\mathcal{J}_d}$,
- $\mathsf{VC}^{\mathcal{I}}_t(e) = \mathsf{VC}^{\mathcal{J}^+_d}_t(e)$ for all $e \in \Delta^{\mathcal{J}_d}$,
- $\mathcal{J}^+_d \not\models \{q_{d,i} \mid i \leq m\}$, and
- $|\Delta^{\mathcal{J}_d}| \leq (|\mathsf{VC}^{\mathcal{I}}_t(d)| + 1)!$.

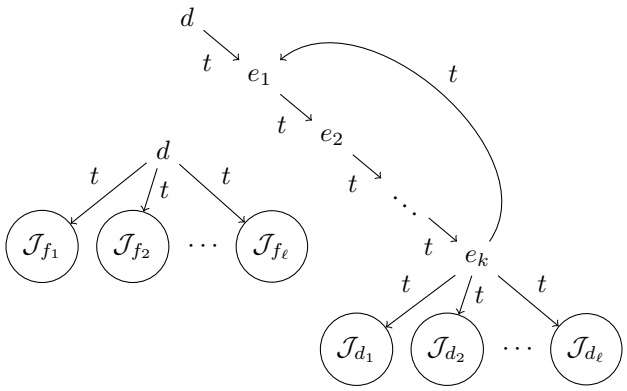

Figure 3: Interpretation $\mathcal{J}_d$ in the two cases.

The second and third condition together immediately give $\mathcal{J}^+_d \models \mathcal{T}$, because $\mathcal{T}$ is in normal form and only uses the role name t. The fourth states that $\mathcal{J}^+_d$ satisfies the same invariant (1) as $\mathcal{I}_d$. In consequence, defining $\mathcal{J}$ as $\mathcal{J}^+_{d_0}$ will complete the proof of Theorem 2.

We construct $\mathcal{J}_d$ for $d \in \Delta^{\mathcal{I}}$ by induction over the set $M = \mathsf{VC}^{\mathcal{I}}_t(d) \subseteq \mathsf{N_C}(\mathcal{T})$. There are two cases, depending on the set $\alpha_M = \left\{e \in \Delta^{\mathcal{I}_d} \mid \mathsf{VC}^{\mathcal{I}}_t(e) = M\right\}$. The first case is conceptually simpler and serves also as the induction base.

1. There exists $e \in \alpha_M$ **such that** $\alpha_M \cap t^{\mathcal{I}}(e) = \emptyset$**.** Hence, $\mathsf{VC}^{\mathcal{I}}_t(e) = M$ but $\mathsf{VC}^{\mathcal{I}}_t(f) \subsetneq M$ for all $f \in t^{\mathcal{I}}(b)$. Then, we can select elements $f_1, \ldots, f_\ell \in t^{\mathcal{I}}(e)$, $\ell \leq |M|$, such that for each $A \in M$, there exists j with $f_j \in A^{\mathcal{I}}$. We build $\mathcal{J}_a$ by taking the disjoint union of interpretations $\mathcal{J}_{f_j}$, that exist by the induction hypothesis, and adding element d with unary type inherited from $\mathcal{I}$ along with t-edges from d to f_j for all j, as shown in Figure 3 (left). Note that in the induction base, where $M = \emptyset$, $\mathcal{J}_d$ has domain $\{d\}$ and no edges.

2. For all $e \in \alpha_M$**,** $\alpha_M \cap t^{\mathcal{I}}(e) \neq \emptyset$**.** Then, because $\preceq$ is a well-founded partial order, there is some $e_0 \in \alpha_M$ and a $\preceq$-minimal m-tuple $\bar{q}$ such that $\bar{q}_e = \bar{q}$ for all $e \in t^{\mathcal{I}}(e_0) \cap \alpha_M$. Since $e_0 \in \alpha_M$, we can, as in the first case, select elements $f_1, \ldots, f_m \in t^{\mathcal{I}}(b_0)$, $m \leq |M|$, such that for each $A \in M$, there is some j with $f_j \in A^{\mathcal{I}}$. We partition $f_1, \ldots, f_m$ in two sequences, depending on $\mathsf{VC}^{\mathcal{I}}_t$:

- $d_1 \ldots, d_\ell$ contains all those f_i with $\mathsf{VC}^{\mathcal{I}}_t(f_i) \subsetneq M$, and
- $e_1, \ldots, e_k$ contains all those f_i with $\mathsf{VC}^{\mathcal{I}}_t(f_i) = M$.

To build $\mathcal{J}_a$, we first arrange $a, e_1, e_2, \ldots, e_k$ (with unary types inherited from $\mathcal{I}$) into a simple path with t-edges. Next, we turn it into a cycle by adding a t-edge from e_k to e_1. Finally, we add the disjoint union of interpretations $\mathcal{J}_{d_1}, \ldots, \mathcal{J}_{d_\ell}$, along with a t-edge from e_k to d_j for all $j \leq \ell$. See Figure 3 (right) for an illustration of the constructed $\mathcal{J}_d$.

Verifying that $\mathcal{J}_d$ has the desired properties is easy, except for $\mathcal{J}^+_d \not\models \{q_{d,i} \mid i \leq m\}$ which subtly relies on the choice of e_0 and the anti-monotonicity of $\preceq$.

5 Pseudo-Tree Queries Differently

To prepare for multi-role queries, we revise the notion of pseudo-tree queries (PTQs). The key property missing in (Eiter et al. 2009) is global undirected acyclicity, which we build into our definition. Intuitively, we define a PTQ as a connected CQ whose set of binary atoms can be partitioned into disjoint connected acyclic sets of atoms over the same role name, called *clusters*, that are arranged into a tree. The latter condition ensures global undirected acyclicity. Let us make this precise.

Definition 3. *Let $q(\bar{x})$ be a CQ. For a non-transitive role name r, an* r-cluster *of $q(\bar{x})$ is a nonempty maximal subset C_r of q of the form $\{r(x, y_1), r(x, y_2), \ldots, r(x, y_k)\}$ with $k > 0$. For a transitive role name t, a* t-cluster *of $q(\bar{x})$ is a nonempty maximal connected set C_t of t-atoms of $q(\bar{x})$.*

The clusters of a CQ constitute a partition of the set of its binary atoms. We treat clusters as (Boolean) conjunctive queries. In particular, we speak of initial variables in clusters. For instance, Figure 4 (left) shows an example of a query with one t-cluster and three s-clusters; cluster C_4 has two initial variables, x and u; variables y and z are not initial in C_4, but they are initial in C_2 and C_3, respectively.

Definition 4. *A* cluster tree *for a CQ $q(\bar{x})$ is a tree having for its set of nodes the set of clusters of $q(\bar{x})$, and such that:*

(a) two clusters can only share variables if they are siblings or if one is a child of the other;
(b) two siblings can only share a variable if they also share it with their parent;
(c) each non-root cluster C shares with its parent exactly one variable, called the entry variable *of C, and this variable must be initial in C (the root cluster has no entry variable).*

Figure 4 (middle) shows a cluster tree for the query on the left. Intuitively, a cluster tree reflects which clusters must be matched below each other, when the query is matched in a transitive-tree interpretation. However, this intuition breaks down as soon as the entry variable of a child cluster is initial in the parent cluster (as for C_1 and C_4 in Figure 4): then, the child cluster need not be matched below the parent cluster.

For similar reasons, cluster trees are not unique: the root of a cluster tree may be swapped for any of its children whose entry variable is initial in the current root. In Figure 4, an alternative cluster tree is obtained by making C_4 a child of C_1. Once we fix the root, the cluster tree is unique: all clusters sharing a variable with the root become its children, etc. By a *root cluster in $q(\bar{x})$* we mean any cluster that is the root of *some* cluster tree for $q(\bar{x})$. In Figure 4, C_1 and C_4 are root clusters, while C_2 and C_3 are not.

Definition 5. *A* Boolean pseudo-tree query (Boolean PTQ) *is a connected Boolean CQ q such that*

1. there is a cluster tree for q;
2. for each transitive t, every t-cluster of q is acyclic.

A unary pseudo-tree query (unary PTQ) *is a unary CQ $q(x)$ such that q is a Boolean PTQ, and x is initial and belongs to a single, root cluster in q. A Boolean/unary UPTQ is a UCQ that contains only Boolean/unary PTQs.*

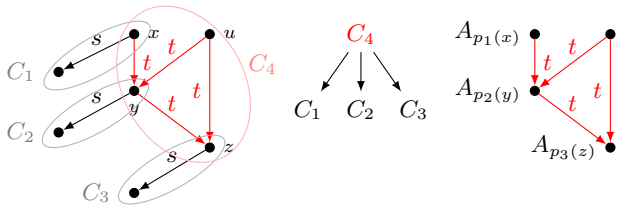

Figure 4: A Boolean PTQ over a transitive role t and non-transitive role s, its cluster tree, and a query corresponding to its root cluster.

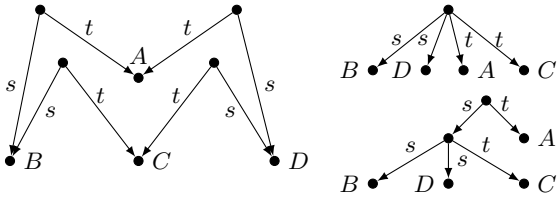

Figure 5: A naughty query using transitive roles s, t.

Coming back to the example in Figure 4, the query q shown on the left is a Boolean PTQ and $q(u)$ is a unary PTQ, whereas $q(x)$, $q(y)$, and $q(z)$ are not. Moreover, any TQ whose answer variable occurs in at most one binary atom (as in Theorem 8 below) is a unary PTQ. On the other hand, the query in Figure 5 (left) does not admit a cluster tree. Yet, it was classified as a PTQ in (Eiter et al. 2009).

The *raison d'être* of PTQs is to semantically capture CQs over transitive-tree interpretations, as stated in Lemma 6 below. Crucially, this is possible only if at most one transitive role is allowed.

Lemma 6. *Let $\mathbf{T}$ be the class of transitive-tree interpretations. The following can be done in polynomial time for CQs using at most one transitive role:*

- *Given a connected Boolean CQ q, decide if $\mathcal{I} \models q$ for some $\mathcal{I} \in \mathbf{T}$, and if so, output a Boolean PTQ $\hat{q}$ such that $\mathcal{I} \models q$ iff $\mathcal{I} \models \hat{q}$ for all $\mathcal{I} \in \mathbf{T}$.*
- *Given a connected unary CQ $q(x)$, decide if $\langle \mathcal{I}, d\rangle \models q(x)$ for some $\mathcal{I} \in \mathbf{T}$ with root d, and if so, output unary PTQs $\hat{q}_1(x), \ldots, \hat{q}_k(x)$ such that for all $\mathcal{I} \in \mathbf{T}$ with root d, $\langle \mathcal{I}, d\rangle \models q(x)$ iff $\langle \mathcal{I}, d\rangle \models \hat{q}_i(x)$ for all i.*

The existence of $\hat{q}$ and $\hat{q}_1(x), \ldots, \hat{q}_k(x)$ in Lemma 6 relies on the input query using at most one transitive role. For instance, the query q in Figure 5 (left) admits a match in a transitive-tree interpretation, but one can check that it is not captured by a single PTQ. Intuitively, this is because q is entailed by both queries in Figure 5 (right), despite their radically different structure. In the unary case, we need multiple (but polynomially many) PTQs, because the answer variable cannot belong to multiple clusters. For instance, for q in Figure 4, $q(x)$ requires two unary PTQs: one consists of cluster C_1 and the other consists of clusters C_2, C_3, and C_4.

We close the section by defining subPTQs, which are to PTQs what subtrees are to TQs. A *subPTQ of a Boolean PTQ q* is a unary subquery of q induced by a proper subtree

of a cluster tree for q. It consists of all binary atoms in the subtree along with all unary atoms of q over variables used in the subtree, and its answer variable is the entry variable of the root of the subtree. A *subPTQ of a unary PTQ* $q(x)$ is defined analogously, except that we are limited to the (unique) cluster tree of $q(x)$ such that x belongs to the root cluster. For instance, the query q in Figure 4 has 4 subPTQs. Three are induced by the subtrees of the cluster tree in the figure, rooted at C_1, C_2, and C_3, and their answer variables are x, y, and z, respectively. The last one is induced by the subtree rooted at C_4 of the alternative cluster tree whose root is C_1, and its answer variable is x. Of those 4 queries, only the first 3 are subPTQs of the unary PTQ $q(u)$.

6 Upper Bounds

Our proof of 2EXPTIME-hardness of CQ entailment in $\mathcal{S}$ crucially relies on the availability of (a) two transitive roles and (b) Boolean CQs. We now show that entailment becomes easier if we forbid either of these. In this sense our hardness result is optimal.

Theorem 7. *The query entailment problem in $\mathcal{S}$ is* coNEXPTIME*-complete for rooted UCQs and for UCQs that use at most one transitive role name.*

The coNEXPTIME-hardness for UCQs using at most one transitive role was established by (Eiter et al. 2009). While the proof is formulated using Boolean CQs, it does not rely on this and can be adjusted easily to the case of rooted CQs.

We focus on the upper bounds. The proofs have a common three-step structure described in the introduction and ultimately establish a *small witness property* based on Theorem 2. We implement Steps 1–3 for both upper bounds in parallel in Sections 6.1–6.3 and we put them together in Section 6.4, where the proof of Theorem 7 is finalized.

6.1 Eliminating ABoxes and Simplifying Queries

The first step eliminates the ABox and simplifies the queries. For the rooted case, this is routine. We reduce our main entailment problem to a variant where the input consists of a TBox $\mathcal{T}$, a set $\tau \subseteq \mathsf{N_C}(\mathcal{T})$, and a unary UCQ Q^1: the task is to decide whether for each model $\mathcal{I}$ of $\mathcal{T}$ and every $d \in \Delta^{\mathcal{I}}$ with $\mathrm{tp}(\mathcal{I}, d) \cap \mathsf{N_C}(\mathcal{T}) = \tau$, we have $\langle \mathcal{I}, a\rangle \models Q^1$. We write this condition as $\langle \mathcal{T}, \tau\rangle \models Q^1$. The reduction is provided by the following theorem. Note that this is a non-deterministic reduction, similar to the one introduced in (Adleman and Manders 1977).

Theorem 8. *There is a* NEXPTIME *algorithm that, given a KB $\langle \mathcal{T}, \mathcal{A}\rangle$, a rooted UCQ $Q(\bar{x})$, and a tuple $\bar{a}$ of individuals from $\mathsf{N_I}(\mathcal{A})$, computes for each $a \in \mathsf{N_I}(\mathcal{A})$ a set $\tau_a \subseteq \mathsf{N_C}(\mathcal{T})$ and a UTQ Q^1_a such that*

- *$\langle \mathcal{T}, \mathcal{A}\rangle \not\models Q(\bar{a})$ iff there is a run of the algorithm such that $\langle \mathcal{T}, \tau_a\rangle \not\models Q^1_a$ for all $a \in \mathsf{N_I}(\mathcal{A})$;*
- *for each run of the algorithm and each $a \in \mathsf{N_I}(\mathcal{A})$, the size of each TQ in Q^1_a is linear in $\|Q\|$ and its answer variable occurs in at most one binary atom.*

Next, we handle the single transitive role case. We obtain stronger size guarantees (needed later) at the cost of relaxing tree queries (TQs) to pseudo-tree queries (PTQs). Similarly to the rooted case, we reduce entailment of UCQs using a single transitive role to a variant of entailment for UPTQs. This time, given a TBox $\mathcal{T}$, a set $\tau \subseteq \mathsf{N_C}(\mathcal{T})$, a Boolean UPTQ Q^0, and a unary UPTQ Q^1, one has to decide whether for each model $\mathcal{I}$ of $\mathcal{T}$ and each $d \in \Delta^{\mathcal{I}}$ with $\mathrm{tp}(\mathcal{I}, d) \cap \mathsf{N_C}(\mathcal{T}) = \tau$, we have $\mathcal{I} \models Q^0$ or $\langle \mathcal{I}, d\rangle \models Q^1$. We write the condition to be decided as $\langle \mathcal{T}, \tau\rangle \models Q^0 \vee Q^1$. The following theorem provides the reduction; it relies on Lemma 6 to transform CQs into PTQs equivalent over transitive-tree interpretations.

Theorem 9. *There is a* NEXPTIME *algorithm that, given a KB $\langle \mathcal{T}, \mathcal{A}\rangle$ and a Boolean UCQ Q using at most one transitive role, computes for each $a \in \mathsf{N_I}(\mathcal{A})$ a set $\tau_a \subseteq \mathsf{N_C}(\mathcal{T})$, a Boolean UPTQ Q^0_a, and a unary UPTQ Q^1_a such that*

- *$\langle \mathcal{T}, \mathcal{A}\rangle \not\models Q$ iff there is a run of the algorithm such that $\langle \mathcal{T}, \tau_a\rangle \not\models Q^0_a \vee Q^1_a$ for all $a \in \mathsf{N_I}(\mathcal{A})$;*
- *for each run of the algorithm and each $a \in \mathsf{N_I}(\mathcal{A})$, $\|Q^0_a\|$ is linear in $\|Q\|$, each PTQ in Q^1_a is linear in $\|Q\|$, and the total number of subPTQs of PTQs from Q^1_a is polynomial in $\|Q\|$.*

Compared to Theorem 8, Theorem 9 yields queries from a broader class (UPTQs, rather than UTQs), but offers stronger size guarantees: the total number of subPTQs of PTQs in Q^1_a is polynomial. This is a substitute for a polynomial bound on $\|Q^1_a\|$, which cannot be ensured. Importantly, the variant of entailment used in Theorem 8 is a special case of the one in Theorem 9: $Q^0_a = \emptyset$ and UTQs with answer variables used in at most one binary atom instead of UPTQs. This enables us to treat the two cases in parallel in what follows.

6.2 From Entailment to Existence of Mosaics

Our goal now is to reduce the variant of UPTQ entailment introduced in Section 6.1 to the special case where only one role is used in the input. Our reduction relies on *tiles*, which are interpretations using only one role name, consistent with the TBox. We show that countermodels can be represented using *mosaics*, which are collections of such tiles.

Below, $\mathcal{T}_r$ is the restriction of $\mathcal{T}$ to CIs that do not mention any role names other than r, and $\langle \mathcal{I}, d_0\rangle \vDash \mathcal{T}_r$ means that $d_0 \in C^{\mathcal{I}}$ implies $d_0 \in D^{\mathcal{I}}$ for each CI $C \sqsubseteq D$ from $\mathcal{T}_r$.

Definition 10. *A* tile *for a TBox $\mathcal{T}$ is a triple $\langle \mathcal{I}, d_0, r\rangle$ where $\mathcal{I}$ is an interpretation, $d_0 \in \Delta^{\mathcal{I}}$, and $r \in \mathsf{N_R}$, such that $s^{\mathcal{I}} = \emptyset$ for all $s \in \mathsf{N_R} - \{r\}$ and*

1. *if r is transitive, then $\mathcal{I} \vDash \mathcal{T}_r$ (and, in particular, $\mathcal{I}^+ = \mathcal{I}$);*
2. *if r is non-transitive, then $\langle \mathcal{I}, d_0\rangle \vDash \mathcal{T}_r$.*

Tiles are meant to be assembled to form a countermodel to $\langle \mathcal{T}, \tau\rangle \vDash Q^0 \vee Q^1$. To facilitate this, we introduce a family of fresh auxiliary concept names of the form $A_{p(x)}$, where $p(x)$ is a subPTQ of a PTQ from Q^0 or from Q^1. Intuitively, $A_{p(x)}$ will be used to propagate the constraint that $p(x)$ must not be matched. To prevent the entire CQ from being satisfied, we recursively partition it into clusters (as described in the next paragraph) and ensure that each tile violates suitable clusters, augmented with atoms of the form $A_{p(x)}(x)$. Crucially, if Q^0 and Q^1 are Q^0_a and Q^1_a from Theorem 9 for some Q, then the number of auxiliary concept names is polynomial in $\|Q\|$.

Let q be Boolean PTQ from Q^0 or the Boolean PTQ underlying a subPTQ $q(x)$ of a PTQ from Q^0 or from Q^1. Consider

a root cluster C of q, and the corresponding cluster tree for q with C in the root. We define q_C as the Boolean PTQ obtained by replacing each direct subtree of the cluster tree with a single unary atom using a suitable auxiliary concept name. (Recall that a direct subtree is one rooted at a child of the whole tree's root.) More precisely, q_C contains all binary atoms from C along with all unary atoms of q over variables used in C, and for each child C' of C, sharing a variable x with C, q_C contains the atom $A_{p(x)}(x)$ where $p(x)$ is the subPTQ corresponding to the cluster subtree rooted at C'.

For instance, if q is the query in Figure 4 (left), then $q_{C_4}(x)$ is the query shown on the right. Concept names $A_{p_1(x)}$, $A_{p_2(y)}$, and $A_{p_3(z)}$ used in $q_{C_4}(x)$ replace the subPTQs of q induced by C_1, C_2, and C_3, respectively.

We now define a mosaic as a compatible collection of tiles that correctly propagates information about subPTQs.

Definition 11. *Consider a TBox $\mathcal{T}$, a type τ, a Boolean UPTQ Q^0, and a unary UPTQ Q^1. A* mosaic for $\mathcal{T}$ and τ, and against Q^0 and Q^1, *is a set $\mathfrak{M}$ of tiles for $\mathcal{T}$ such that:*

1. *for each $\langle \mathcal{I}, d_0, r\rangle \in \mathfrak{M}$, $q \in Q^0$, and root r-cluster C of q, we have $\mathcal{I} \not\models q_C$;*
2. *for each $\langle \mathcal{I}, d_0, r\rangle \in \mathfrak{M}$, $d \in \Delta^{\mathcal{I}}$, auxiliary $A_{p(x)}$, and root r-cluster C of p that contains x, if $d \notin A^{\mathcal{I}}_{p(x)}$ then $\langle \mathcal{I}, d\rangle \not\models p_C(x)$;*
3. *there is a family of tiles $\langle \mathcal{I}_r, d_r, r\rangle \in \mathfrak{M}$ with $\mathrm{tp}(\mathcal{I}_r, d_r) \cap \mathsf{N_C}(\mathcal{T}) = \tau$, for r ranging over $\mathsf{N_R}(\mathcal{T})$, such that for each $q(x) \in Q^1$, $\langle \mathcal{I}_r, d_r\rangle \not\models q_C(x)$ for some $r \in \mathsf{N_R}(\mathcal{T})$ and root r-cluster C of q that contains x;*
4. *for each $\langle \mathcal{I}, d_0, r\rangle \in \mathfrak{M}$, $d \in \Delta^{\mathcal{I}}$, and $s \in \mathsf{N_R}(\mathcal{T})$ such that either $s \neq r$ or both r is non-transitive and $d \neq d_0$, there is $\langle \mathcal{J}, e_0, s\rangle \in \mathfrak{M}$ such that $\mathrm{tp}(\mathcal{I}, d) = \mathrm{tp}(\mathcal{J}, e_0)$.*

As promised, our variant of the entailment problem for UPTQs reduces to the existence of mosaics.

Theorem 12. *For any TBox $\mathcal{T}$, $\tau \subseteq \mathsf{N_C}(\mathcal{T})$, Boolean UPTQ Q^0, and unary UPTQ Q^1, $\langle \mathcal{T}, \tau\rangle \not\models Q^0 \vee Q^1$ iff there is a mosaic $\mathfrak{M}$ for $\mathcal{T}$ and τ, and against Q^0 and Q^1.*

The proof of Theorem 12 is technical yet relatively standard. It remains to see that the existence of suitable mosaics can be decided in NEXPTIME. Towards this end, we prove in the following subsection that tiles and mosaics of singly exponential size are sufficient.

6.3 Bounding the Sizes of Tiles and Mosaics

We know from Theorem 12 that the entailment problem reduces to the existence of a mosaic. Now, using Theorem 2 in Section 4, we can show that we can assume each tile of the mosaic to be of exponential size.

Lemma 13. *Consider a TBox $\mathcal{T}$, $\tau \subseteq \mathsf{N_C}(\mathcal{T})$, a Boolean UPTQ Q^0, and a unary UPTQ Q^1. For every mosaic $\mathfrak{M}$ for $\mathcal{T}$ and τ, and against Q^0 and Q^1, there is a mosaic $\mathfrak{M}'$ for $\mathcal{T}$ and τ against Q^0 and Q^1 such that $|\Delta^{\mathcal{I}}| \leq (|\mathsf{N_C}(\mathcal{T})| + 1)!$ for each tile $\langle \mathcal{I}, a_0, r\rangle \in \mathfrak{M}'$.*

It remains to see that the number of tiles in a mosaic can be bounded as well. Here, the arguments diverge.

In the single transitive role case, we rely on the number of auxiliary concept names, determined by the number of subPTQs of PTQs from Q^0 and Q^1, being polynomial. As one tile per type is enough, we get the following.

Lemma 14. *If there is a mosaic for $\mathcal{T}$ and τ, and against a Boolean UPTQ Q^0 and a unary UPTQ Q^1, with tiles of size at most M, then there is one with at most $n + n \cdot 2^{n+m}$ tiles, all of size at most M, where $n = \|\mathcal{T}\|$ and m is the total number of subPTQs of PTQs from Q^0 and Q^1.*

In the rooted case, the number of auxiliary concepts is exponential, but we observe that in a countermodel built from tiles, a rooted query can traverse only a linear number of tiles from the initial element. Because a tile of size M requires at most $M \cdot \|\mathcal{T}\|$ witnesses, a singly exponential number of tiles is sufficient to build the part of the countermodel within linear distance from the initial element. Further away, we do not care about matching the query anymore, so we only need one tile for each $\tau \subseteq \mathsf{N_C}(\mathcal{T})$.

Lemma 15. *If there is a mosaic for $\mathcal{T}$ and τ against $Q^0 = \emptyset$ and unary UTQ Q^1 with tiles of size at most M, then there is one with at most $n \cdot ((Mn)^{m+1} + 2^n)$ tiles, all of size at most M, where $n = \|\mathcal{T}\|$ and m is the maximal number of variables of a TQ in Q^1.*

6.4 Wrapping Up

Combining Theorems 8 and 9 (Step 1), Theorem 12 (Step 2), and Lemmas 13–15 (Step 3), we obtain the following.

Corollary 16. *There is a* NEXPTIME *algorithm that, given a KB $\langle \mathcal{T}, \mathcal{A}\rangle$ and rooted UCQ Q or a Boolean UCQ Q using at most one transitive role, computes for each $a \in \mathsf{N_I}(\mathcal{A})$ a set $\tau_a \subseteq \mathsf{N_C}(\mathcal{T})$, a Boolean PTQ Q^0_a and a unary UPTQ Q^1_a, both containing PTQs of linear size only, such that $\langle \mathcal{T}, \mathcal{A}\rangle \not\models Q$ iff there is a run of the algorithm such that for all $a \in \mathsf{N_I}(\mathcal{A})$ there is a mosaic $\mathfrak{M}_a$ for $\mathcal{T}$ and τ_a against Q^0_a and Q^1_a, of size bounded exponentially in $\|\mathcal{T}\| + \|Q\|$ using tiles of size bounded exponentially in $\|\mathcal{T}\|$.*

Theorem 7 now follows easily, because after computing Q^0_a and Q^1_a for each $a \in \mathsf{N_I}(\mathcal{A})$, the algorithm from Corollary 16 can guess a suitable mosaic $\mathfrak{M}_a$ for each a. Verifying that $\mathfrak{M}_a$ is indeed a mosaic for $\mathcal{T}$ and τ_a against Q^0_a and Q^1_a can be done in time exponential in $\|\mathcal{T}\| + \|Q\|$.

7 Conclusions

Contrary to previous expectations, UCQ entailment in $\mathcal{S}$ turns out to be 2EXPTIME-complete, even when restricted to trivial ABoxes and CQs with two transitive roles. On the positive side, both entailment of rooted UCQs and entailment of UCQs using at most one transitive role are CONEXPTIME-complete and thus easier. We note a curious dependence of the complexity of the problem on the number of transitive roles allowed in queries: EXPTIME for 0, CONEXPTIME for 1, and 2EXPTIME for at least 2.

Our results partly apply to UCQ entailment over finite interpretations. Indeed, it is easy to check that our 2EXPTIME-hardness proof also works in the finite case. A matching upper bound follows from (Gogacz, Ibáñez-García, and Murlak 2018). On the other hand, it is an open question if our CONEXPTIME upper bounds hold in the finite case, too.

Acknowledgments

Vincent Michielini was supported by the ERC grant INFSYS, agreement no. 950398, held by Wojciech Czerwiński at the University of Warsaw. Filip Murlak was supported by Poland's NCN grant 2018/30/E/ST6/00042.

References

Adleman, L.; and Manders, K. 1977. Reducibility, randomness, and intractibility (Abstract). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, 151–163. New York, NY, USA: Association for Computing Machinery. ISBN 9781450374095.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press. ISBN 978-0-521-69542-8.

Bednarczyk, B. 2024. Data Complexity in Expressive Description Logics with Path Expressions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024*, 3241–3249. ijcai.org.

Bednarczyk, B.; and Rudolph, S. 2023. How to Tell Easy from Hard: Complexities of Conjunctive Query Entailment in Extensions of ALC. *J. Artif. Intell. Res.*, 78.

Bienvenu, M.; Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2010. Query Answering in the Description Logic S. In *Proceedings of the 23rd International Workshop on Description Logics (DL 2010)*, volume 573 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2014. Answering regular path queries in expressive Description Logics via alternating tree-automata. *Inf. Comput.*, 237: 12–55.

Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2009. Query Answering in Description Logics with Transitive Roles. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, 759–764.

Glimm, B.; Horrocks, I.; and Sattler, U. 2008. Unions of Conjunctive Queries in SHOQ. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 252–262. AAAI Press.

Glimm, B.; Lutz, C.; Horrocks, I.; and Sattler, U. 2008. Conjunctive Query Answering for the Description Logic SHIQ. *J. Artif. Intell. Res.*, 31: 157–204.

Gogacz, T.; Ibáñez-García, Y. A.; and Murlak, F. 2018. Finite Query Answering in Expressive Description Logics with Transitive Roles. In *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning KR 2018*, 369–378. AAAI Press.

Gottlob, G.; Pieris, A.; and Tendera, L. 2013. Querying the Guarded Fragment with Transitivity. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7966 of *Lecture Notes in Computer Science*, 287–298. Springer.

Gutiérrez-Basulto, V.; Ibáñez-García, Y.; Jung, J. C.; and Murlak, F. 2023. Answering regular path queries mediated by unrestricted SQ ontologies. *Artif. Intell.*, 314: 103808.

Ibáñez-García, Y. A.; Jung, J. C.; Michielini, V.; and Murlak, F. 2025. Revisiting Conjunctive Query Entailment for $\mathcal{S}$. arXiv:2511.07933.

Lutz, C. 2008. The Complexity of Conjunctive Query Answering in Expressive Description Logics. In *Proceedings of 4th International Joint Conference on Automated Reasoning, IJCAR 2008*, volume 5195 of *Lecture Notes in Computer Science*, 179–193. Springer.

Ortiz, M.; Simkus, M.; and Eiter, T. 2008. Worst-case Optimal Conjunctive Query Answering for an Expressive Description Logic without Inverses. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, 504–510. AAAI Press.

Rosati, R. 2011. On the finite controllability of conjunctive query answering in databases under open-world assumption. *Journal of Computer and System Sciences*, 77(3): 572–594.