Succinctness of Query Rewriting in OWL 2 QL: The Case of Tree-like Queries

Meghyn Bienvenu1, Stanislav Kikot2, and Vladimir Podolskii3

1 LRI – CNRS & Université Paris Sud, Orsay, France
2 Institute for Information Transmission Problems & MIPT, Moscow, Russia
3 Steklov Mathematical Institute, Moscow, Russia

Abstract. This paper further investigates the succinctness landscape of query rewriting in OWL 2 QL. We clarify the worst-case size of positive existential (PE), non-recursive Datalog (NDL), and first-order (FO) rewritings for various classes of tree-like conjunctive queries, ranging from linear queries up to bounded treewidth queries. More specifically, we establish a superpolynomial lower bound on the size of PE-rewritings that holds already for linear queries and TBoxes of depth 2. For NDL-rewritings, we show that polynomial-size rewritings always exist for tree-shaped queries with a bounded number of leaves (and arbitrary TBoxes), and for bounded treewidth queries and bounded depth TBoxes. Finally, we show that the succinctness problems concerning FO-rewritings are equivalent to well-known problems in Boolean circuit complexity. Along with known results, this yields a complete picture of the succinctness landscape for the considered classes of queries and TBoxes.

1 Introduction

For several years now, conjunctive query (CQ) answering has been a major focus of description logic (DL) research (cf. survey [19]), due to the growing interest in using description logic ontologies to query data. Formally, the problem is to compute the certain answers to a CQ $q(\mathbf{x})$ over a knowledge base $(\mathcal{T}, \mathcal{A})$, that is, the tuples of individuals $\mathbf{a}$ that satisfy $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$. Much of the work on CQ answering focuses on lightweight DLs of the DL-Lite family [5], and the corresponding OWL 2 QL profile [18]. The popularity of these languages is due to fact that they enjoy first-order (FO) rewritability, which means that for every CQ $q(\mathbf{x})$ and every TBox $\mathcal{T}$, there exists a computable FO-query $q'(\mathbf{x})$ (called a rewriting) such that the certain answers to $q(\mathbf{x})$ over $(\mathcal{T}, \mathcal{A})$ coincide with the answers of the FO-query $q'(\mathbf{x})$ over the ABox $\mathcal{A}$ (viewed as a database). First-order rewritability provides a means of reducing CQ answering to the evaluation of FO ($\sim$ SQL) queries in relational databases. A great many different query rewriting algorithms have been proposed for OWL 2 QL and its extensions, cf. [5, 20, 25, 6, 10, 24, 21, 7, 17, 23]. Most of these algorithms produce rewritings expressed as unions of conjunctive queries (UCQs), and the size of such rewritings can be huge, making it difficult, or even impossible, to evaluate them using standard relational database management systems.

It is not difficult to see that exponential-size rewritings are unavoidable if rewritings are given as UCQs (consider for instance the CQ $q(\mathbf{x}) = B_1(\mathbf{x}) \land \ldots \land B_n(\mathbf{x})$ and TBox...
A natural (and non-trivial) question is whether an exponential blowup can be avoided by moving to other standard query languages, like positive existential (PE) queries, non-recursive datalog (NDL) queries, or first-order (FO-) queries. More generally, under what conditions can we ensure polynomial-size rewritings? A first (negative) answer was given in [14], which proved exponential lower bounds for the worst-case size of PE- and NDL-rewritings, as well as a superpolynomial lower bound for FO-rewritings (under the widely-held assumption that \( \text{NP} \not\subseteq \text{P/poly} \)). Interestingly, all three results hold already for tree-shaped CQs, which are a well-studied class of CQs that often enjoy better computational properties, cf. [29, 4]. While the queries used in the proofs had a simple structure, the TBoxes induced full binary trees of depth \( n \). This raised the question of whether better results could be obtained by considering restricted classes of TBoxes. A recent study [15] explored this question for TBoxes of depth 1 and 2, that is, TBoxes that generate canonical models whose elements are at most 1 or 2 ‘steps away’ from the ABox (see Section 2 for a formal definition). It was shown that for depth 1 TBoxes, polysize PE-rewritings do not exist, polysize NDL-rewritings do exist, and polysize FO-rewritings exist iff \( \text{NL/poly} \subseteq \text{NC} \). For depth 2 TBoxes, neither polysize PE- nor NDL-rewritings exist, and polysize FO-rewritings do not exist unless \( \text{NP} \not\subseteq \text{P/poly} \). These results used simpler TBoxes, but the considered CQs were no longer tree-shaped. For depth 1 TBoxes, this distinction is crucial, as it was further shown in [15] that polysize PE-rewritings do exist for tree-shaped CQs.

---

4 We focus on so-called pure FO-rewritings, cf. for [8, 11] discussion and related results.
While existing results go a long way towards understanding the succinctness landscape of query rewriting in OWL 2 QL (cf. Figure 1, two leftmost clouds and two clouds labelled NP/poly), a number of questions remain open:

- What happens if we consider tree-shaped queries and bounded depth TBoxes?
- What happens if we consider generalizations or restrictions of tree-shaped CQs?

In this paper, we address these questions by providing a complete picture of the succinctness of rewritings for tree-shaped queries, their restriction to linear and bounded branching queries (i.e., tree-shaped CQs with a bounded number of leaves), and their generalization to bounded treewidth queries. More specifically, we establish a super-polynomial lower bound on the size of PE-rewritings that holds already for linear queries and TBoxes of depth 2. For NDL-rewritings, we show that polynomial-size rewritings always exist for bounded branching queries (and arbitrary TBoxes), and for bounded treewidth queries and bounded depth TBoxes. Finally, we show that the succinctness problems concerning FO-rewritings are equivalent to well-known problems in Boolean circuit complexity: NL/poly ⊆ NC¹ in the case of linear and bounded branching queries, and SAC¹ ⊆ NC¹ in the case of tree-shaped and bounded treewidth queries and bounded depth TBoxes. Our new results give us the two middle clouds in Figure 1 and, together with known results, yield a complete picture of the succinctness landscape for the considered classes of queries and TBoxes. To prove our results, we establish tight connections between Boolean functions induced by queries and TBoxes and the non-uniform complexity classes NL/poly and SAC¹, reusing and further extending the machinery developed in [14, 15].

2 Preliminaries

OWL 2 QL In this paper, we use the simplified DL syntax of the OWL 2 QL profile [18]. As usual, we assume countably infinite, mutually disjoint sets \( \mathbb{N}_C \), \( \mathbb{N}_R \), and \( \mathbb{N}_I \) of concept names, role names, and individual names. Roles \( R \) and basic concepts \( B \) are defined by the grammar:

\[
R ::= r | r^-
\]

\[
B ::= A | \exists R
\]

where \( A \in \mathbb{N}_C \) and \( r \in \mathbb{N}_R \). We use \( \mathbb{N}_R^\pm \) to refer to the set of all roles.

A TBox (typically denoted \( \mathcal{T} \)) is a finite set of inclusions of the forms

\[
B_1 \sqsubseteq B_2 \quad B_1 \sqsubseteq \neg B_2 \quad R_1 \sqsubseteq R_2 \quad R_1 \sqsubseteq \neg R_2
\]

The signature of a TBox \( \mathcal{T} \), written \( \text{sig}(\mathcal{T}) \), is the set of concept and role names that appear in \( \mathcal{T} \). An ABox (typically denoted \( \mathcal{A} \)) is a finite set of assertions of the form \( A(a) \) or \( r(a, b) \), where \( A \in \mathbb{N}_C \), \( r \in \mathbb{N}_R \), and \( a, b \in \mathbb{N}_I \). The set of individual names in \( \mathcal{A} \) is denoted \( \text{Inds}(\mathcal{A}) \).

A TBox \( \mathcal{T} \) and ABox \( \mathcal{A} \) together form a knowledge base (KB) \( \mathcal{K} = (\mathcal{T}, \mathcal{A}) \). The semantics of KBs is defined in the usual way based on interpretations \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \) [3]. We use \( \sqsubseteq_\mathcal{T} \) to denote the subsumption relation induced by \( \mathcal{T} \) and write \( P_1 \sqsubseteq_\mathcal{T} P_2 \) if \( \mathcal{T} \models P_1 \sqsubseteq P_2 \), where \( P_1, P_2 \) are both concepts or roles.
**Query answering and rewriting** A conjunctive query (CQ) \( q(x) \) is an FO-formula \( \exists y \varphi(x, y) \), where \( \varphi \) is a conjunction of atoms of the form \( A(z_1) \) or \( r(z_1, z_2) \) with \( z_i \in x \cup y \). The free variables \( x \) are called answer variables. Note that we assume w.l.o.g. that CQs do not contain individual names, and where convenient, we regard a CQ as the set of its atoms. We use \( \text{vars}(q) \) (resp. \( \text{avars}(q) \)) to denote the set of variables (resp. answer variables) of \( q \). The signature of \( q \), denoted \( \text{sig}(q) \), is the set of concept and role names in \( q \). To every CQ \( q \), we associate the undirected graph \( G_q \) whose vertices are the variables of \( q \), and which contains an edge \( \{u, v\} \) whenever \( q \) contains an atom \( r(u, v) \) or \( r(v, u) \). We call a CQ \( q \) tree-shaped if the graph \( G_q \) is a tree\(^5\).

A tuple \( a \subseteq \text{Inds}(A) \) is a certain answer to \( q(x) \) over \( K = (T, A) \) if \( I \models q(a) \) for all \( I \models K \); in this case we write \( K \models q(a) \). By first-order semantics, \( I \models q(a) \) iff there is a mapping \( h : \text{vars}(q) \to \Delta^T \) such that (i) \( h(z) \in A^I \) whenever \( A(z) \in q \), (ii) \( (h(z), h(z')) \in r^I \) whenever \( r(z, z') \in q \), and (iii) \( h \) maps \( \text{avars}(q) \) to \( a^I \). If the first two conditions are satisfied, then \( h \) is a homomorphism from \( q \) to \( I \), and we write \( h : q \to I \). If (iii) also holds, then we write \( h : q(a) \to I \).

To every ABox \( A \), we associate the interpretation \( I_A \) whose domain is \( \text{Inds}(A) \) and whose interpretation function makes true precisely the assertions from \( A \). We say that an FO-formula \( q'(x) \) with free variables \( x \) and without constants is an FO-rewriting of CQ \( q(x) \) and TBox \( T \) if, for any ABox \( A \) and any \( a \subseteq \text{Inds}(A) \), we have \( T, A \models q(a) \) iff \( I_A \models q'(a) \). If \( q' \) is a positive existential formula (i.e. it only uses \( \exists, \land, \lor \)), then it is called a PE-rewriting of \( q \) and \( T \). We also consider rewritings in the form of nonrecursive Datalog queries. We remind the reader that a Datalog program (typically denoted \( \Pi \)) is a finite set of rules \( \forall x (\gamma_1 \land \cdots \land \gamma_m \to \gamma_0) \), where each \( \gamma_i \) is an atom of the form \( P(x_1, \ldots, x_l) \) with \( x_i \in x \). The atom \( \gamma_0 \) is called the head of the rule, and \( \gamma_1, \ldots, \gamma_m \) its body. All variables in the head must also occur in the body. A predicate \( P \) depends on a predicate \( Q \) in program \( \Pi \) if \( \Pi \) contains a rule whose head predicate is \( P \) and whose body contains \( Q \). The program \( \Pi \) is called nonrecursive if there are no cycles in the dependence relation for \( \Pi \). For a nonrecursive Datalog program \( \Pi \) and an atom \( \text{goal}(x) \), we say that \( (\Pi, \text{goal}) \) is an NDL-rewriting of \( q(x) \) and \( T \) in case \( T, A \models q(a) \) iff \( \Pi, A \models \text{goal}(a) \), for any ABox \( A \) and any \( a \subseteq \text{Inds}(A) \).

For \( \forall \in \{\text{PE, NDL, FO}\} \), we say that queries from \( \Omega \) and TBoxes from \( \Sigma \) have polysize \( \forall \)-rewritings if there exists a polynomial \( p \) such that every \( q \in \Omega \) and \( T \in \Sigma \) has a \( \forall \)-rewriting \( q' \) with \( |q'| \leq p(|q| + |T|) \).

**Canonical model** We recall that every consistent OWL 2 QL KB \( (T, A) \) possesses a canonical model \( C_{T,A} \) with the property that \( T, A \models q(a) \) iff \( C_{T,A} \models q(a) \), for every CQ \( q \) and tuple \( a \subseteq \text{Inds}(A) \). The domain of \( C_{T,A} \) consists of all individual names from \( A \) and all sequences \( a R_1 R_2 \ldots R_n \) \((n \geq 1) \) such that

1. \( T, A \models \exists R_1(a) \);
2. for every \( 1 \leq i < n \): \( T \models \exists R_i^\top \subseteq \exists R_{i+1} \) and \( T \not\models R_i^\bot \subseteq R_{i+1} \).

---

\(^5\) Tree-shaped conjunctive queries also go by the name of acyclic queries, cf. [29, 4]
Concept and role names are interpreted as follows:

\[
A^C_{T,A} = \{a \in \text{Inds}(A) \mid T, A \models A(a)\} \cup \{wR \in \Delta^C_{T,A} \mid T \models \exists R^+ \subseteq A\}
\]

\[
r^C_{T,A} = \{(a, b) \mid r(a, b) \in A\} \cup \{(w, wS) \in \Delta^C_{T,A} \times \Delta^C_{T,A} \mid T \models S \subseteq r\} \cup
\]

\[
\{(wS, w) \in \Delta^C_{T,A} \times \Delta^C_{T,A} \mid T \models S \subseteq r^{-}\}
\]

Every individual name \(a \in \text{Inds}(A)\) is interpreted as itself: \(a^C_{T,A} = a\).

We say that a TBox \(T\) is of depth \(\omega\) if there is an ABox \(A\) such that \(C_{T,A}\) has an infinite domain; \(T\) is of depth \(d\), \(0 \leq d < \omega\), if \(d\) is the greatest number such that some \(C_{T,A}\) contains an element of the form \(aR_1 \ldots R_d\). The depth of \(T\) can be computed in polynomial time, and if \(T\) is of finite depth, then its depth cannot exceed \(2|T|\).

### 3 Boolean Functions as a Tool for Studying Rewritings

In this section, we introduce different representations of Boolean functions that will play an important role in our results. We assume that the reader is familiar with Boolean circuits [1, 12], built using AND, OR, and NOT gates. The size of a circuit \(C\), denoted \(|C|\), is defined as its number of gates. We will be particularly interested in monotone circuits (that is, circuits with no NOT gates). (Monotone) formulas are (monotone) circuits whose underlying graph is a tree.

Non-deterministic branching programs (NBP) are another well-known model for the representation of Boolean functions [22, 12]. An NBP is defined as a tuple \(P = (V, E, s, t, I)\), where \((V, E)\) is an directed graph, \(s, t \in V\), and \(I\) is a function that labels every edge \(e \in E\) with a conjunction of propositional literals. The NBP \(P\) induces the function \(f_P\) defined as follows: for every valuation \(\alpha\) of the propositional variables in \(P\), \(f_P(\alpha) = 1\) if and only if there is a path from \(s\) to \(t\) in the graph \((V, E)\) such that all labels along the path evaluate to 1 under \(\alpha\).

#### 3.1 Hypergraph functions and hypergraph programs

We recall hypergraph functions and programs from [15]. Let \(H = (V, E)\) be a hypergraph with vertices \(v \in V\) and hyperedges \(e \in E, \ e \subseteq 2^V\). A subset \(E' \subseteq E\) is independent if \(e \cap e' = \emptyset\), for any distinct \(e, e' \in E'\). With each vertex \(v \in V\) and each hyperedge \(e \in E\), we associate propositional variables \(p_v\) and \(p_e\), respectively. The hypergraph function \(f_H\) for \(H\) is given by the Boolean formula

\[
f_H = \bigvee_{\text{ind. } E' \subseteq E} \left( \bigwedge_{v \in V \cup E'} p_v \land \bigwedge_{e \in E'} p_e \right).
\]

**Example 1.** Consider the hypergraph \(H = (V, E)\) with \(V = \{v_1, v_2, v_3\}\), \(E = \{e_1, e_2\}\) where \(e_1 = \{v_1, v_2\}\) and \(e_2 = \{v_2, v_3\}\). It gives rise to the hypergraph function

\[
f_H = \left( p_{v_1} \land p_{v_2} \land p_{v_3} \right) \lor \left( p_{e_1} \land p_{v_3} \right) \lor \left( p_{e_2} \land p_{v_1} \right),
\]

whose three disjuncts correspond to the three independent sets \(I_1 = \emptyset\), \(I_2 = \{e_1\}\), and \(I_3 = \{e_2\}\) of \(E\).
A hypergraph program HGP \( P \) consists of a hypergraph \( H_P = (V, E) \) and a function \( l_P \) that labels every vertex with 0, 1, \( p_i \) or \( \neg p_i \) (here the \( p_i \) are propositional variables, distinct from the \( p_v, p_e \) above). An input for \( P \) is a valuation of the propositional variables in \( P \)'s labels. We say that the hypergraph program \( P \) computes a Boolean function \( f \) in case, for any input \( \alpha \), we have \( f(\alpha) = 1 \) if and only if there is an independent subset of \( E \) that covers all zeros—that is, contains all the vertices in \( V \) labelled with 0 under \( \alpha \). A hypergraph program is monotone if there are no negated variables among its vertex labels. The size, \( |H| \), of a hypergraph program \( H \) is the number of its vertices and hyperedges. Observe that each hypergraph program that is based upon the hypergraph \( H \) computes the Boolean function that is obtained from the hypergraph function \( f_H \) by substituting vertex labels for vertex variables and 1 for edge variables. Conversely, it is not hard to construct for a given hypergraph \( H \), a hypergraph program that computes \( f_H \). Sometimes it is convenient to consider hypergraph programs whose labels are conjunctions of variables and their negations, rather than single literals. It is not hard to see that this does not change the power of such programs.

### 3.2 Upper bounds via tree witness hypergraph functions

The upper bounds in [15] rely on associating a hypergraph function with every query and TBox. As the hypergraph is defined in terms of tree witnesses, we first recall the definition of tree witnesses. Consider a CQ \( q \) and TBox. As the hypergraph is defined in terms of tree witnesses, we first recall the definition of tree witnesses. Consider a CQ \( q \) and TBox \( T \). For every role \( R \), we let \( T_R = T \cup \{ A_R \subseteq \exists R \} \) and \( A_R = \{ A_R(a) \} \) (for some fresh concept name \( A_R \)). Suppose that \( q' \subseteq q \) (recall that we view queries as sets of atoms) and there is a homomorphism \( h: q' \to C_{T_R \cap A_R} \) such that \( h(x) = a \) for every \( x \in \text{avars}(q) \). Let \( t_e = \{ z \in \text{vars}(q') \mid h(z) = a \} \), and let \( t_i \) be the remaining set of (quantified) variables in \( q' \). We call the pair \( (t_e, t_i) \) a tree witness for \( q \) and \( T \) generated by \( R \) if \( t_i \neq \emptyset \) and \( q' \) is a minimal subset of \( q \) such that, for any \( y \in t_i \), every atom in \( q \) containing \( y \) belongs to \( q' \). In this case, we denote \( q' \) by \( q_t \). Note that the same tree witness can be generated by different roles \( R \). We let \( \Theta_H^q \) be the set of all tree witnesses of \( q \) and \( T \) and use \( \Theta_H^q[R] \) to denote those generated by \( R \). We use \( z \in t \) as a shorthand for \( z \in t_e \cup t_i \).

To every CQ \( q \) and TBox \( T \), we can naturally associate the hypergraph whose vertices are the atoms of \( q \) and whose hyperedges are the sets \( q_t \), for tree witnesses \( t \) for \( q \) and \( T \). We denote this hypergraph by \( H_T^q \) and call the corresponding function \( f_{H_T^q} \) the tree witness hypergraph function of \( q \) and \( T \).

It is known that the circuit complexity of \( f_{H_T^q} \) provides an upper bound on the size of rewritings of \( q \) and \( T \).

**Theorem 1 (from [15]).** If \( f_{H_T^q} \) is computed by a (monotone) Boolean formula \( \chi \) then there is a (PE-) FO-rewriting of \( q \) and \( T \) of size \( O(|\chi| \cdot |q| \cdot |T|) \).

If \( f_{H_T^q} \) is computed by a monotone Boolean circuit \( C \) then there is an NDL-rewriting of \( q \) and \( T \) of size \( O(|C| \cdot |q| \cdot |T|) \).

The key observation for the proof of Theorem 1 is that a rewriting of \( q \) and \( T \) may be obtained by substituting query atoms for the corresponding vertex variables and special “tree witness formulas” (which express that the tree onto which tree witness is mapped is actually present in the ABox) for edge variables in a formula which computes \( f_{H_T^q} \).
Example 2. Consider the query
\[
q(x_1, x_4) = \exists y_2, y_3 \left( r_1(x_1, y_2) \land r_2(y_2, y_3) \land r_3(y_3, x_4) \right)
\]
and the TBox \( T \) which consists of the following axioms:
\[
A \subseteq \exists s \quad B \subseteq \exists t \quad s \subseteq r_1 \quad t \subseteq r_2 \quad s \subseteq r_2^- \quad t \subseteq r_3^-
\]
This query has two tree witnesses \( t^1 = (t^1_1, t^1_2) \) with \( t^1_1 = \{x_1, y_3\} \) and \( t^1_2 = \{y_2\} \) and \( t^2 = (t^2_1, t^2_2) \) with \( t^2_1 = \{x_4, y_2\} \) and \( t^2_2 = \{y_3\} \). The corresponding subqueries \( q_{t^1} \) and \( q_{t^2} \) are shown as rectangles in the following picture. They can be mapped to the trees on the left, which are generated, for example, by unary concept names \( A \) and \( B \).

The tree witness hypergraph \( H^q_T \) has vertices \( (r_1(x_1, y_2), r_2(y_2, y_3), \) and \( r_3(y_3, x_4) \) and two edges: \( q_{t^1} = \{(r_1(x_1, y_2), r_2(y_2, y_3))\} \) and \( q_{t^2} = \{(r_2(y_2, y_3), r_3(y_3, x_4))\} \). Observe that \( H^q_T \) is isomorphic to the the hypergraph \( H \) from Example 1, and so it gives rise to the same Boolean function (up to renaming):
\[
f_{H^q_T} = (p_{r_1}(x_1, y_2) \land p_{r_2}(y_2, y_3) \land p_{r_3}(y_3, x_4)) \lor (p_{q_{t^1}} \land p_{q_{t^2}}) \lor (p_{q_{t^2}} \land p_{r_1}(x_1, y_2)).
\]
We can obtain a rewriting \( q' \) of \( q \) and \( T \) by substituting

- \( r_1(x_1, y_2) \lor s(x_1, y_2) \) for \( p_{r_1}(x_1, y_2) \),
- \( r_2(y_2, y_3) \lor s(y_2, y_3) \lor t(y_2, y_3) \) for \( p_{r_2}(y_2, y_3) \),
- \( r_3(y_3, x_4) \lor t(x_4, y_1) \) for \( p_{r_3}(y_3, x_4) \),
- \( x_1 = y_3 \land A(x_1) \) for \( p_{q_{t^1}} \), and
- \( y_2 = x_4 \land B(x_4) \) for \( p_{q_{t^2}} \)

in the formula \( f_{H^q_T} \) and quantifying existentially over \( y_2, y_3 \).

Observe that the function \( f_{H^q_T} \) contains a variable \( p_t \) for every tree witness \( t \). For this reason, it can only be used to show polynomial upper bounds in cases where \( |\Theta^q_T| \) is bounded polynomially in \( |q| \) and \( |T| \).

In order to obtain useful upper bounds even in the case when there may be exponentially many tree witnesses, we consider a slightly different function
\[
f'_{H^q_T} = \bigvee_{\Theta \subseteq \Theta^q_T \text{ independent}} \left( \bigwedge_{e \in q \setminus q_0} p_e \land \left( \bigwedge_{t \in \Theta} \left( \bigwedge_{z, z' \in t} p_{z=z'} \land \bigvee_{R \in \mathbb{N}^*_R} \bigwedge_{z \in t} p^{(R)}_{z} \right) \right) \right) \tag{2}
\]
that is obtained from \( f_{H^q}^T \) by replacing each variable \( p_t \) by

\[
\bigwedge_{z,z' \in t} p_{z=z'} \land \bigvee_{R \in N^C_k, t \in \Theta^T_q[R]} \bigwedge_{z \in t} p^R_z.
\]

Intuitively, the variable \( p_{z=z'} \) enforces that variables \( z \) and \( z' \) are mapped to elements of \( C_{\mathcal{T},A} \) that begin by the same ABox individual; the variable \( p^R_z \) states that \( z \) is mapped to an element that begins by an individual \( a \) satisfying \( \mathcal{T},A \models \exists R(a) \).

\section*{Example 3.}
For \( q \) and \( T \) from the previous example, we have

\[
f^T_{H^q} = (p_{r_1(x_1,y_2)} \land p_{r_2(y_2,y_3)} \land p_{r_3(y_3,x_4)}) \lor
(p_{r_3(y_3,x_4)} \land \bigwedge_{z,z' \in \{x_1,y_2,y_3\}} (p_{z=z'} \land p^S_z)) \lor
(p_{r_1(x_1,y_2)} \land \bigwedge_{z,z' \in \{y_2,y_3,x_4\}} (p_{z=z'} \land p^T_z)).
\]

\( \Box \)

We observe that the number of variables in \( f^T_{H^q} \) is polynomially bounded in \(|q|\) and \(|T|\), but \( f^T_{H^q} \) retains the same properties as \( f_{H^q} \) regarding upper bounds.

\section*{Theorem 2.}
Theorem 1 continues to hold if \( f_{H^q} \) is replaced by \( f^T_{H^q} \).

\section*{Remark.}
In fact, Theorem 1 was proved in [15] only for consistent KBs. But it is known that it is possible to define a short PE-query \( q^+ \) that when evaluated on \( \mathcal{I}_A \) returns all \( k \)-tuples of individual names on \( \mathcal{I}_A \) if the KB \( (\mathcal{T},A) \) is inconsistent, and returns no answers otherwise, cf. [2]. It follows that if we have a query \( q' \) that is a rewriting for \( q \) and \( T \) for all ABoxes \( A \) that are consistent with \( T \), then we can obtain a rewriting for \( q \) and \( T \) (that works for all ABoxes) by taking the disjunction of \( q' \) and \( q^+ \). Therefore, to prove Theorem 2, it sufficient to show how to construct such “consistent rewritings”.

\section*{Proof.}
Let \( q \) be a CQ with answer variables \( x \) and existential variables \( y \) (we will use \( z \) when referring to variables of either type). Consider a Boolean formula \( \chi \) that computes \( f^T_{H^q} \), and let \( q' \) be the FO-formula obtained by replacing \( p_\varphi \) by the atom \( \varphi \), replacing \( p_{z=z'} \) by the equality \( z = z' \), replacing \( p^R_z \) by the standard PE-rewriting \( \rho_R(z) \) of \( \exists R(z) \) and \( T \), and quantifying the existential variables \( y \) of \( q \). Note that \( q' \) has the same answer variables as \( q \), and if \( \chi \) is a monotone formula, then \( q' \) is a PE-formula.

We wish to show that \( q'' \) is a consistent rewriting of \( q \) and \( T \) (cf. preceding remark). To do so, we let \( q'' \) be the PE-formula obtained by applying the above transformation to the original monotone Boolean formula \( f^T_{H^q} \):

\[
q'' = \exists y \bigvee_{\varnothing \subseteq \Theta^T_q} \left( \bigwedge_{e \in q \setminus \Theta_0} e \land \bigwedge_{t \in \Theta^T_q} \bigwedge_{z,z' \in t} z = z' \land \bigvee_{R \in N^C_k, t \in \Theta^T_q[R]} \rho_R(z) \right).
\]
We know that $\chi$ and $f'_{q,T}$ compute the same (monotone) Boolean function. It follows that $q'$ and $q''$ are equivalent FO-formulas. It thus suffices to show that $q''$ is a consistent rewriting of $q$ and $T$. This is easily shown by comparing $q''$ to the following query $q'''$:

$$
\exists y \bigvee_{\theta \subseteq \Theta_T} \left( \bigwedge_{\rho \in q \setminus q_\Theta} \bigwedge_{t \in \Theta} \bigwedge_{R \in \mathbb{N}_T^t} \exists z (\rho_R(z) \land \bigwedge_{z' \in \ell} z' = z) \right)
$$

which was proven in [15] to be a consistent FO-rewriting of $q$ and $T$.

Indeed, by the definition of a tree witness, for any tree witness $t$,

$$\bigwedge_{z,z' \in t} z = z'$$

is equivalent to

$$\bigwedge_{z,z' \in \ell} z = z'$$

as subformulas of $q'''$ since extra equalities contain only existentially quantified variables that do not appear in any atoms (other than equalities) in the corresponding disjuncts. It is also clear that

$$(\bigwedge_{z,z' \in t} z = z') \land \rho_R(z)$$

is equivalent to

$$(\bigwedge_{z,z' \in t} z = z') \land \rho_R(z'')$$

for some $z'' \in t$. \Box

### 3.3 Lower bounds via primitive evaluation functions

In order to obtain lower bounds on the size of rewritings, it will prove convenient to associate to each pair $(q, T)$ a third function $f^P_{q,T}$ that describes the result of evaluating $q$ on single-individual ABoxes. Given Boolean vectors $\alpha : \mathbb{N}_C \cap (\text{sig}(T) \cup \text{sig}(q)) \rightarrow \{0, 1\}$ and $\beta : \mathbb{N}_R \cap (\text{sig}(T) \cup \text{sig}(q)) \rightarrow \{0, 1\}$, we let

$$A(\alpha, \beta) = \{A(a) \mid \alpha(A) = 1\} \cup \{r(a, a) \mid \beta(r) = 1\}$$

and set $f^P_{q,T}(\alpha, \beta) = 1$ iff $T, A(\alpha, \beta) \models q(a)$, where $a$ is a tuple of $a$'s of the required length. We call $f^P_{q,T}$ the primitive evaluation function for $q$ and $T$.

**Theorem 3.** If $q'$ is a (PE-) FO-rewriting of $q$ and $T$, then there is a (monotone) Boolean formula $\chi$ of size $O(|q'|)$ which computes $f^P_{q,T}$.

If $(\Pi, G)$ is an NDL-rewriting of $q$ and $T$, then $f^P_{q,T}$ is computed by a monotone Boolean circuit $C$ of size $O(|\Pi|)$. 


This theorem is implicitly proved in [15]. For the sake of readability, we include an outline of the proof.

**Proof.** Given a PE-, FO- or NDL-rewriting $q'$ of $q$ and $T$, we show how to construct, respectively, a monotone Boolean formula, a Boolean formula or a monotone Boolean circuit for the function $f_{q,T}^P$ of size $|q'|$.

Suppose $q'$ is a PE-rewriting of $q$ and $T$. We eliminate the quantifiers in $q'$ by first replacing every subformula of the form $\exists x \psi(x)$ in $q'$ with $\psi(a)$, and then replacing each atom of the form $A(a)$ and $r(a,a)$ with the corresponding propositional variable. One can verify that the resulting propositional monotone Boolean formula computes $f_{q,T}^P$. If $q'$ is an FO-rewriting of $q$, then we eliminate the quantifiers by replacing $\exists x \psi(x)$ and $\forall x \psi(x)$ in $q'$ with $\psi(a)$. We then proceed as before, replacing atoms $A(a)$ and $r(a,a)$ by the corresponding propositional variables, to obtain a propositional Boolean formula computing $f_{q,T}^P$.

If $(\Pi, \text{goal})$ is an NDL-rewriting of $q$, then we replace all the variables in $\Pi$ with $a$ and then perform the replacement described above. Denote the resulting propositional NDL-program by $\Pi'$. The program $\Pi'$ can now be transformed into a monotone Boolean circuit computing $f_{q,T}^P$. For every (propositional) variable $p$ occurring in the head of a rule in $\Pi'$, we introduce an OR-gate whose output is $p$ and inputs are the bodies of the rules with head $p$; for each such body, we introduce an AND-gate whose inputs are the propositional variables in the body. \qed

## 4 Bounded Branching Queries

It is known from [14] that tree-shaped CQs do not have polysize PE- or NDL-rewritings (nor polysize FO-rewritings, unless NP $\subseteq$ P/poly). In this section, we investigate the robustness of these results by considering a restricted form of tree-shaped queries. We will say that a tree-shaped CQ $q$ has $k$ leaves if the associated graph $G_q$ (defined in Section 2) contains exactly $k$ vertices of degree 1. We will be interested in bounded branching queries (that is, tree-shaped CQs with a bounded number of leaves) and linear queries (having exactly 2 leaves).

### 4.1 Bounded branching queries, interval hypergraphs and NBPs

It is not hard to see that every linear query induces a tree witness hypergraph that is isomorphic to an interval hypergraph, i.e., a hypergraph of the form $H = (V,E)$, where $V = \{[i,i+1] \mid 1 \leq i < n\}$ for some finite $n$ and $E$ is a set of intervals of the form $[i,j]$ where $1 \leq i < j \leq n$. We slightly abuse notation and think of a segment $[i,j]$ as the set of all integer segments of unit length which are inside it; in symbols $[i,j] = \{k, k+1 \mid 1 \leq k < j\}$. Alternatively, interval hypergraphs can be defined in a simpler way as the hypergraphs of the form $(V,E)$ where $V = \{1,2,\ldots,n\}$ and every $e \in E$ is of the form $[i,j] = \{k \in V \mid i \leq k \leq j\}$, but we prefer to think of $V$ consisting of segments of unit length rather than numbers in order to have common notation for linear hypergraphs and tree hypergraphs (introduced in the next section).

One can easily show that the hypergraph functions of interval hypergraphs can be computed by polynomial-size NBPs.

The program $\Pi'$ can now be transformed into a monotone Boolean circuit computing $f_{q,T}^P$. For every (propositional) variable $p$ occurring in the head of a rule in $\Pi'$, we introduce an OR-gate whose output is $p$ and inputs are the bodies of the rules with head $p$; for each such body, we introduce an AND-gate whose inputs are the propositional variables in the body. \qed

## 4 Bounded Branching Queries

It is known from [14] that tree-shaped CQs do not have polysize PE- or NDL-rewritings (nor polysize FO-rewritings, unless NP $\subseteq$ P/poly). In this section, we investigate the robustness of these results by considering a restricted form of tree-shaped queries. We will say that a tree-shaped CQ $q$ has $k$ leaves if the associated graph $G_q$ (defined in Section 2) contains exactly $k$ vertices of degree 1. We will be interested in bounded branching queries (that is, tree-shaped CQs with a bounded number of leaves) and linear queries (having exactly 2 leaves).

### 4.1 Bounded branching queries, interval hypergraphs and NBPs

It is not hard to see that every linear query induces a tree witness hypergraph that is isomorphic to an interval hypergraph, i.e., a hypergraph of the form $H = (V,E)$, where $V = \{[i,i+1] \mid 1 \leq i < n\}$ for some finite $n$ and $E$ is a set of intervals of the form $[i,j]$ where $1 \leq i < j \leq n$. We slightly abuse notation and think of a segment $[i,j]$ as the set of all integer segments of unit length which are inside it; in symbols $[i,j] = \{k, k+1 \mid 1 \leq k < j\}$. Alternatively, interval hypergraphs can be defined in a simpler way as the hypergraphs of the form $(V,E)$ where $V = \{1,2,\ldots,n\}$ and every $e \in E$ is of the form $[i,j] = \{k \in V \mid i \leq k \leq j\}$, but we prefer to think of $V$ consisting of segments of unit length rather than numbers in order to have common notation for linear hypergraphs and tree hypergraphs (introduced in the next section).

One can easily show that the hypergraph functions of interval hypergraphs can be computed by polynomial-size NBPs.
**Theorem 4.** There is a polynomial \( p \) such that for every interval hypergraph \( H = (V^H, E^H) \), there is a monotone NBP \( P \) with \( |P| \leq p(|H|) \) that computes \( f_H \).

**Proof.** Suppose that \( V^H = \{ [i, i+1] \mid 1 \leq i \leq n \} \). Since both vertices and hyperedges of \( H \) are intervals, we denote all the variables of \( f_H \) (both vertices and edges) by \( p_{[i,j]} \). Consider the NBP based on the graph \( G = (V^G, E^G) \), where:

- \( V^G = E^H \cup \{ s, t \} \) with \( s = [0, 0] \) and \( t = [n+1, n+1] \)
- \( E^G \) contains a directed edge from \([i_1, j_1]\) to \([i_2, j_2]\) iff \( i_2 > j_1 \), and this edge is labelled by

\[
\bigwedge_{j_1 \leq k < j_2} p_{[k,k+1]} \wedge p_{[i_2,j_2]}
\]

Note that we assume here that \( p_{[n+1, n+1]} = 1 \). It is easy to see that for any propositional valuation \( \alpha \) of the variables \( p_{[i,j]} \), \( f_H(\alpha) = 1 \) if and only if \( t \) is reachable from \( s \) in \( G \) via a path that is labelled with 1’s under the valuation \( \alpha \).

It follows from Theorem 4 and our observation that linear queries give rise to interval hypergraphs that tree hypergraph functions of linear queries can be computed using polynomial-size NBPs. In fact, we show with the next theorem that this contains to hold if we move to the wider class of bounded branching queries:

**Theorem 5.** Fix a constant \( \ell > 1 \). Then there exists a polynomial \( p \) such that for every tree-shaped CQ \( q \) with at most \( \ell \) leaves and every OWL 2 QL TBox \( T \), there is an NBP of size at most \( p(|q| + |T|) \) that computes \( f_{H^q} \).

**Proof.** We fix \( \ell \), \( q \) and \( T \) and construct an NBP that computes \( f_{H^q} \). Suppose that \( H = (V_H, E_H) \) is the tree witness hypergraph for \( q \) and \( T \). Note that since all tree witnesses are connected and there is a bound on the number of leaves in \( q \), the number of tree witnesses \( |E_H| \) is bounded polynomially in \( |q| \). We choose one of \( q \)'s leaves as a root \( r \) which introduces a partial order on the variables of \( q \), so we have \( \ell - 1 \) remaining leaves. An independent subset \( E' \subseteq E_H \) is called good, if every simple path leading from the root to a remaining leaf intersects at most one edge of \( E' \). Clearly, any good \( E' \) contains at most \( \ell - 1 \) hyperedges and so the number of all good subsets \( E' \) is polynomially bounded the in size of \( H \) (again, we exploit the fact that \( \ell \) is fixed).

For good subsets \( E' \) and \( E'' \), we say that \( E' \) precedes \( E'' \) (written \( E' \prec E'' \)) if any simple path \([r, l] \) (which connects the root \( r \) to a leaf \( l \) ) which passes through \( E'' \), passes through \( E' \) and meets \( E' \) on the side of \( r \) from \( E'' \) (see Fig. 2). For such \( E' \) and \( E'' \), we denote by midlab(\( E', E'' \)) the conjunction of the labels for all vertices outside \( E' \) and \( E'' \) that are accessible from \( E' \) via paths not passing through \( E'' \) and that are not accessible from \( r \) via a path not passing through \( E' \). For a good \( E' \), we denote by predlab(\( E' \)) the conjunction of the labels of all vertices outside \( E' \) that are accessible from \( r \) via paths not passing through \( E' \). By finlab(\( E' \)), we denote the conjunction of the labels of all vertices outside \( E' \) which are accessible from \( r \) only via paths passing through \( E' \).

Now we are ready to construct the NBP based on the graph \( G = (V_G, E_G) \). We set \( V_G = \{ E' \subseteq E_H \mid E' \text{ is good} \} \cup \{ s, t \} \). We connect \( s \) with all \( E' \in V_G \) and label these edges by predlab(\( E' \)). Then we connect \( E' \) to \( E'' \) if \( E' \) precedes \( E'' \) and label
Fig. 2. In this figure $E'$ consists of one edge, $E''$ consists of two edges and both are independent and good. Furthermore, $E'$ precedes $E''$. Now $\text{predlab}(E')$ is the segment between $r$ and $E'$, $\text{midlab}(E', E'')$ comprises the segment between $E'$ and $E''$ and also the branch which goes out of $E'$ downward, $\text{finlab}(E'')$ consist of two branches going out from $E''$ toward leaves.

these edges by $\text{midlab}(E', E'')$. Finally, we connect all $E'$ to $t$ and label such edges by $\text{finlab}(E')$.

We claim that under any valuation $p$, the vertex $t$ is accessible from $s$ if and only if there is an independent subset $\hat{E} \subseteq E_H$ covering all zeros. Indeed, any covering $\hat{E}$ splits into “layers” $E^1, E^2, \ldots, E^m$ which form a path from $s$ to $t$ in $G$: take $E^1$ to be the set of all edges from $\hat{E}$ that are accessible from $r$ via paths which do not pass through any hyperedge of $\hat{E}$; take $E^2$ to be the set of all edges from $\hat{E} \setminus E^1$ which are accessible from $r$ via paths which do not go through any hyperedge of $\hat{E} \setminus E^1$, and so on. Conversely, any path leading from $s$ to $t$ gives us a covering which is the union of all hyperedges that label the vertices on this path.

We next give a polynomial translation from NBPs into interval hypergraph programs, thereby establishing the polynomial equivalence of these formalisms:

**Theorem 6.** Every function $f$ that is computable by an NBP $P$ is also computable by an interval hypergraph program of size polynomial in $|P|$.

To prove Theorem 6, we need the following lemma.

**Lemma 1.** Let $G_e$ be the directed graph on $n$ vertices that has loops around all its vertices, and whose other edges are given by an adjacency matrix $e = \{e_i \mid 1 \leq i \leq m\}$ where $m = n(n-1)$. Then the $v_1$-$v_n$-reachability function $f$ – defined by setting $f(e) = 1$ iff $v_n$ is reachable from $v_0$ in $G_e$ – is computable by a polynomial-size hypergraph program based on an interval hypergraph $H = (V_H, E_H)$.

**Proof.** We give an explicit description of this hypergraph. Its vertices are arranged into $n$ vertex blocks and $n-1$ edge blocks which alternate. Every vertex block contains $2n$ points. The first two points correspond to $v_1$, the next two to $v_2$ and so on. The points in the vertex blocks are denoted by $v_i^h$ and $\bar{v}_i^h$, where $h (1 \leq h \leq n)$ is the number of the block, and $i (1 \leq i \leq n)$ designates the vertex inside the block. Similarly, every edge block contains $m$ pairs of points denoted by $e_i^h, \bar{e}_i^h$, where $h (1 \leq h \leq n-1)$ is...
the number of the block and \( i \) \((1 \leq i \leq M)\) is the number of the edge inside the block. We have \( M = m + n \) to include loops which are \( e_i \)'s for \( m < i \leq M \). We assume that \( e_i = 1 \) for \( m < i \leq M \).

We remove the first and the last vertices \( v_1^1 \) and \( v_n^n \). These points are linearly ordered as follows:

\[
\begin{array}{cccccccccc}
1 & 2 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & n
\end{array}
\]

We assume that \( v_1 \) is the first vertex in the block and \( i \) \((1 \leq i \leq M)\) is the number of the edge inside the block.

The set of vertices \( V_H \) corresponds to the unit intervals in this order:

\[
V_H = \{ \{v^h_{i_1}, v^h_{i_2}\} | 1 \leq h \leq n, 1 \leq i_1 \leq n\} \cup \{\{v^h_{i_1}, v^h_{i_2}\} | 1 \leq h \leq n, 1 < i_1 \leq n\} \cup \{\{e^h_i, e^{h+1}_i\} | 1 \leq h < n\} \cup \{\{v^h_n, e^{h+1}_n\} | 1 \leq h < n\} \cup \{\{v^1_1, v^1_2\}, \{v^n_n, v^n_{n+1}\}\}
\]

Now we set

\[
E_H = \bigcup_{e_i = (v_j, v_k)} \left( \bigcup_{h=1}^{n-1} \{ [v^h_j, e^h_i], [e^h_i, v^h_{i+1}] \} \right),
\]

where \([u, u']\) are segments of the depicted linear order and \(i\) ranges from 1 to \(M\). The label function \( f \) marks \( [e^h_i, e^{h+1}_i] \) by \( e_i \) and all other vertices (= unit segments) by 0. Thus \([e^h_i, e^{h+1}_i]\) for \(m < i \leq M\) are marked by 1.

It is easy to see that under each assignment \( e \), the vertex \( v_n \) is accessible from \( v_1 \) in \( G_e \) iff there is an independent subset \( E'\) of \( E_H \) covering all zeros. Indeed, suppose that \( v_n \) is reachable from \( v_1 \). Then because of the loops in \( G \), it is reachable in exactly \( n \) steps, so there is path \( v_{i_1}, v_{i_2}, \ldots, v_{i_n} \) in \( G \) with \( i_1 = 1 \) and \( i_n = n \). In this case, we can take

\[
E' = \bigcup_{h=1}^{n-1} \{ [v^h_{i_h}, e^h_{\#(i, i_{h+1})}], [e^h_{\#(i, i_{h+1})}, v^h_{i_{h+1}}] \},
\]

where \(\#(i, j)\) is the number of the edge which leads from \(v_i\) to \(v_j\). Conversely, one can show that any independent set which covers all zeros is of this form, and so it gives us a path \(v_{i_1}, v_{i_2}, \ldots, v_{i_n} \) in \( G \) with \( i_1 = 1 \) and \( i_n = n \).

**Proof (of Theorem 6).** Take an NBP \( P = (G, s, t, l) \). First, note that adding loops around any vertex of \( G \) does not change the function computed by \( P \) so without loss of generality, we may assume that \( G \) has loops around each vertex. Now note that \( f_P \) is the result of a substitution into the \( s\)-t-reachability function \( f \) of \( G_e \) (that is, we can compute \( f_P \) by first evaluating the labels of the edges under the given valuation and using the result to define the adjacency matrix \( e \)). Now we apply Lemma 1 to obtain a hypergraph program \( P' \) for \( f \). By performing the aforementioned substitution on the labels in \( P' \), we get a hypergraph program for \( f_P \).

---

\(^6\) To make the construction more readable, we use the points \( v^h_1, v^h_i, e^h_i, \) and \( e^h \). Since these points are linearly ordered, they could be replaced by a sequence of natural numbers \(1, 2, \ldots\) to conform to our definition of interval hypergraphs.
To complete the chain, we show how to compute hypergraph functions for interval hypergraphs using primitive evaluation functions of linear CQs and TBoxes of depth 2. To every interval hypergraph $H$, we associate the linear CQ $q_H$:

$$q_H = \exists y \bigwedge_{[i, i+1] \in V} (r_i(y_i, y'_i) \land r'_i(y'_i, y_{i+1})),$$

and the TBox $T_H$ that contains the following axioms for each edge $[i, j] \in E$:

$$B_{ij} \subseteq \exists s_{ij} \quad \exists s'_{ij} \subseteq \exists s'_{ij} \quad s_{ij} \subseteq r_i \quad s'_{ij} \subseteq r'_j$$

$$\{s'_{ij} \subseteq r'_k \mid i < k < j\} \quad \{s'_{ij} \subseteq r_k \mid i < k \leq j\}$$

**Example 4.** Consider the interval hypergraph $H = (V, E)$ where $V = \{[i, i+1] \mid 1 \leq i \leq 3\}$ corresponds to the interval $[1, 4]$ and $E$ contains a single hyperedge $\{[1, 2], [2, 3]\}$ corresponding to the interval $[1, 3]$. The preceding construction yields the query $q_H$ pictured here:

![Diagram of interval hypergraph](image)

and the TBox $T_H$ consisting of the following axioms:

$$B_{13} \subseteq \exists s_{13}, \quad \exists s'_{13} \subseteq \exists s'_{13}, \quad s_{13} \subseteq r_1, \quad s'_{13} \subseteq r'_2$$

$$s'_{13} \subseteq r'_1, \quad s'_{13} \subseteq r_2$$

On the right, we display the portion of the canonical model generated by $B_{13}$. Observe how the subquery of $q_H$ lying between $y_1$ and $y_2$ can be mapped onto it.

**Theorem 7.** For every interval hypergraph $H = (V, E)$ and for all $\alpha : V \to \{0, 1\}$ and $\beta : E \to \{0, 1\}$ we have $f_H(\alpha, \beta) = 1$ iff $f_{q_H, T_H}(\gamma)$ with $\gamma$ defined as follows: $\gamma(B_{ij}) = \beta([i, j])$, $\gamma(r_i) = \gamma(r'_i) = \alpha([i, i+1])$, and $\gamma(s_{ij}) = \gamma(s'_{ij}) = 0$.

**Proof.** ($\Rightarrow$) Suppose that $f_{q_H, T_H}(\gamma) = 1$, and let $h : q_H \to C_{T_H, \mathcal{A}(\alpha, \beta)}$ be a homomorphism. Then the set $E' = \{[i, j] \mid h(y'_i) \neq a s_{ij}\}$ forms an independent subset of $E$. Indeed, suppose for a contradiction that there were intervals $[i, j]$ and $[i', j']$ in $E'$ that overlap (i.e. $i \leq i' < j$). Since $h(y'_i) = a s_{ij}$, the structure of $T_H$ ensures that $h(y'_i) = a s_{ij}$ for every $1 \leq i \leq k$, and in particular, $h(y'_i) = a s_{ij}$. But that means that we have both $h(y'_i) = a s_{ij}$ and $h(y'_i) = a s'_{ij}$, a contradiction. To see why $E'$ covers all zeros, observe that if $[i, i+1]$ is not covered by $E'$, then $y_i$, $y'_i$, and $y_{i+1}$ must be mapped to $a$, and so we must have $\alpha([i, i+1]) = 1$.

($\Leftarrow$) Suppose that $f_H(\alpha, \beta) = 1$, i.e. there is an independent subset $E' \subseteq E$ covering all zeros. Then we define a homomorphism $h : q_H \to C_{T_H, \mathcal{A}(\alpha, \beta)}$ by setting

$$h(y'_k) = \begin{cases} a s_{ij} & \text{if } k \in [i, j] \text{ for some } [i, j] \in E' \\ a, & \text{otherwise,} \end{cases}$$
and
\[ h(y_k) = \begin{cases} \alpha s_{ij} s'_{ij} & \text{if } i < k < j \text{ for some } [i,j] \in E', \\ \alpha, & \text{otherwise.} \end{cases} \]

It follows that \( T_H, A(\alpha, \beta) \models q_H \), so \( f_{q_H, T_H}^P(\gamma) = 1 \).

### 4.2 Size of Rewritings of Bounded Branching Queries

We now apply the results from Section 4.1 to derive bounds on rewriting size. It is known that there is a sequence \( f_n \) of monotone Boolean functions that are computable by polynomial-size monotone NBPs, but all monotone Boolean formulas computing \( f_n \) are of size \( n^{\Omega(\log n)} \) [13]. Using this fact, together with Theorems 1, 3, 6, and 7, we obtain a strong negative result for PE-rewritings.

**Theorem 8.** There is a sequence of linear CQs \( q_n \) and TBoxes \( T_n \) of depth 2, both of poly-size in \( n \), such that any PE-rewriting of \( q_n \) and \( T_n \) is of size \( n^{\Omega(\log n)} \).

**Proof.** Apply Theorem 1 to the sequence \( f_n \) mentioned above to obtain a sequence of interval hypergraph programs \( P_n \) based on interval hypergraphs \( H_n \) which compute the functions \( f_n \). By Theorem 7, there exist linear CQs \( q_n \) and ontologies \( T_n \) of depth 2 such that \( f_{H_n}^P \) is a subfunction of \( f_{q_n, T_n}^P \). By the construction, \( q_n \) and \( T_n \) are of polynomial size in \( n \). Since \( f_n \) is obtained from \( f_{q_n, T_n}^P \) through a simple substitution, the lower bound \( n^{\Omega(\log n)} \) still holds for \( f_{q_n, T_n}^P \). It remains to apply Theorem 3 to transfer this lower bound to PE-rewritings of \( q_n \) and \( T_n \).  

We obtain a positive result for NDL-rewritings using Theorems 1 and 5 and the fact that NBPs are representable as polynomial-size monotone circuits [22].

**Theorem 9.** Fix a constant \( \ell > 1 \). Then all tree-shaped CQs with at most \( \ell \) leaves and arbitrary TBoxes have polynomial-size NDL-rewritings.

**Proof.** Fix \( \ell > 1 \). By Theorem 6, there exists a polynomial \( p \) such that for every tree-shaped CQ \( q \) with at most \( \ell \) leaves and every TBox \( T \), there is an NBP of size at most \( p(|q| + |T|) \) that computes \( f_{H_q}^P \). We also know from [22] that there is a polynomial \( p' \) such that every function \( f_P \) given by an NBP \( P \) can be computed by a monotone Boolean circuit \( C_P \) of size at most \( p'(P) \). By composing these two translations, we obtain polysize monotone Boolean circuits that compute the functions \( f_{H_q}^P \), for the class of tree-shaped CQs with at most \( \ell \) leaves. It then remains to apply Theorem 1.

Finally, we use Theorems 1, 3, 6, and 7 to show that the existence of polysize FO-rewritings is equivalent to the open problem of whether NL/poly \( \subseteq \) NC\(^1\).

**Theorem 10.** The following are equivalent:

1. There exist polysize FO-rewritings for all linear CQs and depth 2 TBoxes;
2. There exist polysize FO-rewritings for all tree-shaped CQs with at most \( \ell \) leaves and arbitrary TBoxes (for any fixed \( \ell \));
3. There exists a polynomial function \( p \) such that every NBP of size at most \( s \) is computable by a formula of size \( p(s) \). Equivalently, \( \text{NL/poly} \subseteq \text{NC}^1 \).

**Proof.**

(2) \( \implies \) (1): Trivial.

(1) \( \implies \) (3): Suppose (1) holds. In other words, there exists a polynomial \( p \) such that any linear query \( q \) and a TBox \( T \) of depth 2 have a rewriting of the size \( p(|q| + |T|) \).

Consider a sequence of functions \( f_n \) computing s-t-reachability in directed graphs, which is known to be \( \text{NL/poly} \) complete under \( \text{NC}^1 \)-reductions [28] (This function takes the adjacency matrix of an undirected graph \( G \) on \( n \) vertices with two distinguished vertices \( s \) and \( t \) and returns 1 iff \( t \) is accessible from \( s \) in \( G \).) Clearly, the functions \( f_n \) are computed by a sequence of polynomial-size NBPs \( P_n \). Theorem 1 gives us a sequence of hypergraph programs \( P_n' \) based on interval hypergraphs \( H_n \) which compute the \( f_n \). Consider the CQs \( q_{H_n} \) and TBoxes \( T_{H_n} \). By assumption, they have \( \text{PE-rewritings} \) \( q'_{H_n} \) of \( p(|q| + |T|) \) which is polynomial in \( n \). Theorem 3 gives us a polysize monotone Boolean formula for computing \( f_{H_n} \). By Proposition 7 and the definition of the hypergraph program, \( f_n \) is obtained from \( f_{q_{H_n}, T_{H_n}} \) by some substitution. Therefore we have a polysize formula for \( f_n \).

(3) \( \implies \) (2): Suppose (3) holds. Fix some \( \ell \), and take a tree-shaped query \( q \) with at most \( \ell \) leaves and a TBox \( T \). Since \( \ell \) is fixed, by Theorem 5, there is a polysize NBP \( P \) which computes \( f_{H_q} \). By assumption, there is a polysize FO formula computing \( f_{H_q} \), and Theorem 1 transforms it into a FO-rewriting of \( q \) and \( T \).

\[ \square \]

**5 Bounded Treewidth Queries**

In Section 4, we gave bounds on the size of rewritings for restricted classes of tree-shaped queries and their natural generalization to bounded treewidth queries [9]. As the rewriting size of tree-shaped queries and arbitrary TBoxes has already been studied [14], we will focus on a class of "well-behaved" TBoxes, that includes TBoxes of bounded depth as a special case. We begin by formally introducing the classes of queries and TBoxes we consider.

**Bounded treewidth queries** We recall that a tree decomposition of an undirected graph \( G = (V, E) \) is a pair \( (T, \lambda) \) such that \( T \) is an (undirected) tree and \( \lambda \) assigns a label \( \lambda(N) \subseteq V \) to every node \( N \) of \( T \) such that the following conditions are satisfied:

1. For every \( v \in V \), there exists a node \( N \) with \( v \in \lambda(N) \).
2. For every edge \( e \in E \), there exists a node \( N \) such that \( e \subseteq \lambda(N) \).
3. For every \( v \in V \), the nodes \( \{ N \mid v \in \lambda(N) \} \) induce a connected subtree of \( T \).

The width of a tree decomposition \( (T, \lambda) \) is equal to \( \max_N |\lambda(N)| - 1 \), and the treewidth of a graph \( G \) is the minimum width over all tree decompositions of \( G \). The treewidth of a CQ \( q \) is defined as the treewidth of the graph \( G_q \).

**Polynomial image property** Let \( T \) be an OWL 2 QL TBox, and let \( q \) be a CQ. Then the set \( W_{q,T} \) of relevant words for \( q \) and \( T \) consists of all words \( w \) of length at most \( |T| + |q| \) such that there exists an ABox \( A \) that is consistent with \( T \) and a homomorphism
Lemma 2. If $\mathcal{A}$ is consistent with $\mathcal{T}$ and $\mathcal{T}, \mathcal{A} \models q(\alpha)$, then there is some $h : q(\alpha) \rightarrow C_{\mathcal{T}, \mathcal{A}}$ whose image is contained in $\{aw \mid a \in \text{Inds}(\mathcal{A}), w \in W_{q, \mathcal{T}}\}$.

We say that a class $\mathcal{T}$ of TBoxes has the polynomial image property if there is a polynomial $p$ such that for every TBox $\mathcal{T} \in \mathcal{T}$ and every CQ $q$, $|W_{q, \mathcal{T}}| \leq p(|\mathcal{T}| + |q|)$. Observe that if $d \geq 0$ is fixed, then the class of TBoxes of depth at most $d$ has the polynomial image property. Another relevant class of TBoxes with this property is the class of TBoxes that do not contain role inclusions.

5.1 Bounded treewidth queries and tree hypergraph programs

As in Section 4, our first step will be to relate Boolean functions induced by the query and TBox with hypergraph programs. The main difference is that in lieu of interval hypergraph programs, we will use tree hypergraph programs.

To formally define tree hypergraph programs, we must first introduce some definitions related to trees. Given a tree $T$ with vertices $u$ and $v$, the interval $\langle u, v \rangle$ is the set of edges that appear on the simple path connecting $u$ and $v$. If $v_1, \ldots, v_k$ are vertices of $T$, then the generalized interval $\langle v_1, \ldots, v_k \rangle$ is defined as the union of intervals $\langle v_i, v_j \rangle$ over all pairs $(i, j)$. A hypergraph $H = (V_H, E_H)$ is a tree hypergraph if there is a tree $T = (V_T, E_T)$ such that $V_H = E_T$ and every hyperedge in $E_H$ is a generalized interval of $T$. A hypergraph program is a tree hypergraph program (TreeHGP) if it is based on a tree hypergraph.

From $f'_{Hq}$ to TreeHGP. We show how to construct a TreeHGP that computes $f'_{Hq}$, given a TBox $\mathcal{T}$, a CQ $q$, and a tree decomposition $(T, \lambda)$ of $G_q$ of width $t$. We may suppose w.l.o.g. that $T$ contains at most $(2|q| - 1)^2$ nodes, cf. [16]. In order to more easily refer to the variables in $\lambda(N)$, we construct functions $\lambda_1, \ldots, \lambda_t$ such that $\lambda_i(N) \in \lambda(N)$ and $\lambda(N) = \cup_i \lambda_i(N)$.

The basic idea underlying the construction is as follows: for each node $N$ in the tree decomposition of $q$, we select an abstract description of the way the variables in $\lambda(N)$ are homomorphically mapped into the canonical model, and we check that the selected descriptions respect the subqueries of each node and are consistent with each other. Formally, these abstract descriptions are given by the set $\Gamma_i(q, T)$ consisting of all $t$-tuples $w = (w_1, \ldots, w_t)$ of words from $W_{q, \mathcal{T}}$. Intuitively, the words in $w$ specify, for each variable $x$ in $\lambda(N)$, the path of roles that lead from the ABox to the image of $x$ in the canonical model. We say that $w \in \Gamma_i(q, T)$ is consistent with a node $N$ in $T$ if:

- if $A(\lambda_i(N)) \in q$, then either $w[i] = \varepsilon$ or $w[i] = w'R$ and $\exists R^{-} \sqsubseteq_T A$
- if $r(\lambda_i(N), \lambda_j(N)) \in q$, then one of the following holds:
  - $w[i] = w[j] = \varepsilon$
  - $w[j] = w[i] \cdot R$ with $R \sqsubseteq_T r$
  - $w[i] = w[j] \cdot R$ with $R \sqsubseteq_T r^{-}$

We call a pair of tuples $(w_1, w_2)$ compatible with the pair of nodes $(N_1, N_2)$ if:
\( \lambda_i(N_i) = \lambda_j(N_j) \) implies that \( w[i] = w[j] \)

We assume that the elements of \( \Gamma_i(q, T) \) are numbered from 1 to \( M \), and use \( \xi_i \) to refer to the \( i \)-th element. We define a tree \( T' \) that replaces each edge \( \{N_i, N_j\} \) in \( T \) by the following sequence of edges:

\[
\{N_i, u_{i1}\}, \{u_{i1}, u_{i2}\}, \{u_{i2}, v_{i1}\}, \{v_{i1}, v_{i2}\}, \ldots \{u_{ij}, v_{ij}\}, \{v_{ij}, v_{ij+1}\}, \ldots \{u_{iM}, v_{iM}\}, \{v_{iM}, v_{ij}\}, \{v_{ij}, N_j\}
\]

The desired TreeHGP \( (H_{q,T}, q, T) \) is based upon \( T' \) and contains the hyperedges:

\[
E^h_k = \{u_{k1}, \ldots, u_{km}\}, \quad \text{for every } \xi_k \in \Gamma_i(q, T) \text{ that is consistent with } N_i, \quad \text{where } N_{i1}, \ldots, N_{in} \text{ are the neighbours of } N_i
\]

\[
E^{lm}_{ij} = \{v_{ij}, v_{ij+1}\}, \quad \text{for every pair of tuples } (\xi_k, \xi_m) \text{ that is compatible with } (N_i, N_j)
\]

Vertices of the hypergraph (i.e. the edges in \( T' \)) are labeled by \( q_{-T} \) as follows:

\[
\text{every edge of the form } \{N_i, u_{i1}\}, \{v_{ij}, u_{ij+1}\}, \text{ or } \{v_{ij}, v_{ij+1}\} \text{ is labelled } 0
\]

\[
\text{every edge } \{u_{ij}, v_{ij}\} \text{ with } \xi = w \text{ is labelled by the conjuction of:}\n\]

\[
\begin{align*}
&\text{if } \vars(\eta) \subseteq \lambda(N_i) \text{ and } \lambda_g(N_i) \in \vars(\eta) \text{ implies } w[\eta] = \varepsilon \\
&\text{if } \vars(\eta) = \{z\} \subseteq \lambda(N_i), z = \lambda_g(N_i), \text{ and } w[\eta] = Rw' \\
&\text{if } \vars(\eta) = \{z, z'\} \subseteq \lambda(N_i), z = \lambda_g(N_i), z' = \lambda_g'(N_i), \text{ and } w[\eta] = Rw' \text{ or } w'[\eta] = Rw'
\end{align*}
\]

**Theorem 11.** For every TBox \( T \) and CQ \( q \), the TreeHGP \( (H_{q,T}, q, T) \) computes \( f'_{H,q} \).

If \( q \) has treewidth \( t \), then \( |H_{q,T}| \leq 8 \cdot |q|^3 \cdot |W_{q,T}|^2t^2 \).

**Proof.** Let \( (T, \lambda) \) be the tree decomposition of \( G_q \) that of width \( t \) that was used to construct the TreeHGP \( (H_{q,T}, q, T) \). We recall that the formula \( f'_{H,q} \) takes the form

\[
f'_{H,q} = \bigvee_{\Theta \subseteq \Theta_q} \left( \bigwedge_{\eta \in q} p_\eta \wedge \bigwedge_{t \in \Theta} \left( \bigwedge_{z,z' \in t} p_{z,z'} \wedge \bigvee_{R \in N_{q}\_T} \bigwedge_{z \in t} p_{R,z} \right) \right)
\]

where \( q_\Theta = \bigcup_{\Theta} q_{\Theta} \). Throughout the proof, we use \( f_P \) to denote the function computed by the TreeHGP \( (H_{q,T}, q, T) \). Note that by definition \( f_P \) uses exactly the same set of propositional variables as \( f'_{H,q} \).

To show the first direction of the first statement, let \( v \) be a valuation of the variables in \( f'_{H,q} \) such that \( f'_{H,q}(v) = 1 \). Then we can find an independent subset \( \Theta \subseteq \Theta_T \) such that \( v \) satisfies the corresponding disjunct of \( f'_{H,q} \):

\[
\bigwedge_{\eta \in q} p_\eta \wedge \bigwedge_{t \in \Theta} \left( \bigwedge_{z,z' \in t} p_{z,z'} \wedge \bigvee_{R \in N_{q}\_T} \bigwedge_{z \in t} p_{R,z} \right)
\]

For every \( t \in \Theta \), we let \( R_t \) be a role that makes the final disjunction hold. Furthermore, we choose some homomorphism \( h_t : q_{\Theta_t} \to C_{T_{R_t} A_{R_t}} \), where \( T_{R_t} = T \cup \{A_{R_t} \subseteq \exists R_t\} \) and \( A_{R_t} = \{A_{R_t}(a)\} \). Such homomorphisms are guaranteed to exist by the definition of tree witnesses.

Now for each node \( N \) in the tree decomposition \( T \), we define \( w_N \) by setting:
- $w_N[j] = \epsilon$ if $\lambda_j(N) = z$ and either $z$ appears in an atom $q$ such that $v(p_q) = 1$ or there is some $t \in \Theta$ such that $z \in t$.
- $w_N[j] = w$ if $\lambda_j(N) = z$ and there is some $t \in \Theta$ such that $z \in t$, and $h_t(z) = aw$.

First note that $x_N$ is well-defined since the independence of $\Theta$ guarantees that every variable in $q$ can appear in $t$, for at most one tree witness $t \in \Theta$. Moreover, every variable in $q$ must either belong to an atom $q$ such that $v(p_q) = 1$ or to an atom that is contained in $q_t$ for some $t \in \Theta$.

Next we show that $w_N$ is consistent with the node $N$. To show that the first condition holds, consider some atom $A(\lambda_i(N)) \in q$ such that $w_N[i] = \epsilon$. Then there must be a tree witness $t \in \Theta$ such that $\lambda_i(N) \in t_i$, in which case we have that $w_N[i] \neq \epsilon$ and $h_t(\lambda_i(N)) = aw_N[i]$. Let $S$ be the final role in $w_N[i]$. Then since $h_t$ is a homomorphism from $q_t$ into $C_{R_1, A_{R_1}}$, it must be the case that $\exists S^- \subseteq_T A$. To show the second condition holds, consider some atom $r(\lambda_i(N), \lambda_j(N)) \in q$ such that either $w_N[i] \neq \epsilon$ or $w_N[j] \neq \epsilon$. We suppose w.l.o.g. that $w_N[i] \neq \epsilon$ (the other case is handled analogously). It follows from the definition of $w_N$ that there must exist a tree witness $t \in \Theta$ such that $\lambda_i(N) \in t_i$ and $h_t(\lambda_i(N)) = aw_N[i]$. Since $\lambda_j(N) \in t_i$ and $r(\lambda_i(N), \lambda_j(N)) \in q$, the definition of tree witnesses ensures that $\lambda_j(N) \in t$. Because $h_i$ is a homomorphism from $q_t$ into $C_{R_1, A_{R_1}}$, we know that one of the following must hold:

- $\lambda_j(N) \in t_r$, $w_N[j] = \epsilon$, and $w_N[i] = S$ for some $S^- \subseteq_T r$
- $\lambda_j(N) \in t_i$ and $w_N[i] = w_N[j] \cdot S$ for some $S^- \subseteq_T r$
- $\lambda_j(N) \in t_i$ and $w_N[j] = w_N[i] \cdot S$ for some $S \subseteq_T r$

This establishes the second consistency condition holds.

We must also show that the pairs associated with different nodes in $T$ are compatible. To this end, consider a pair of nodes $N_1$ and $N_2$ and the corresponding tuples of words $w_{N_1}$ and $w_{N_2}$. It is clear from the way we defined $w_{N_1}$ and $w_{N_2}$ that if $\lambda_i(N_1) = \lambda_j(N_2)$, then we must have $w_{N_1}[i] = w_{N_2}[j]$.

Now consider the set $E'$ of hyperedges in $H_{q,T}$ that contains:

- for every $N_i$ in $T$, the hyperedge $E_i^k = \langle u_{ij1}^k, \ldots, u_{ijn}^k \rangle$, where $k$ is such that $\xi_k = w_{N_i}$, and $N_{j1}, \ldots, N_{jn}$ are the neighbours of $N_i$.
- for every pair of adjacent nodes $N_i, N_j$ in $T$, the hyperedge $E_{ij}^{km} = \langle v_{ij}^k, v_{ji}^m \rangle$, where $k$ and $m$ are such that $\xi_k = w_{N_i}$ and $\xi_m = w_{N_j}$.

Note that the aforementioned hyperedges all belong to $H_{q,T}$ since we showed that each $w_{N_i}$ is consistent with node $N_i$, and that $w_{N_i}$ and $w_{N_j}$ are compatible with $(N_i, N_j)$ for all pairs of nodes $(N_i, N_j)$ in $T$. It is easy to see that $E'$ is independent, since whenever we include $E_i^k$ or $E_{ij}^{km}$, we do not include any $E_i^{k'}$ or $E_{ij}^{km'}$ for $k' \neq k$.

To see why $E'$ covers all zeros, consider a vertex $F$ of $H_{q,T}$ (= an edge in $T'$) that evaluates to 0 under v. There are several cases to consider:

- $F = \{N_i, u_{ij}^1\}$: $F$ is covered by the hyperedge in $E'$ of the form $E_i^k$
- $F = \{v_{ij}^\ell, u_{ij}^{\ell+1}\}$: then $F$ is either covered by the hyperedge in $E'$ of the form $E_i^k$ (if $k \leq \ell + 1$) or by the hyperedge $E_{ij}^{km}$ (if $k > \ell + 1$)
- $F = \{v_{ij}^M, v_{ji}^M\}$: then $F$ is covered by the hyperedge in $E'$ of the form $E_{ij}^{km}$.
Since all zeros are covered, we know that for every node \( k \) be the case that
\[ \{H_i \} \]
decompositions, we can infer that the pairs assigned to the consistency and compatibility properties, and the connectedness condition of tree tent with
\[ N \]
We have thus shown that
\[ F = \{ u^f_{i,j}, v^f_{i,j} \} \]
with \( \xi_\ell = w \): then \( F \) is either covered by the hyperedge in \( E' \) of the form \( E^k_i \) (if \( \ell < k \)) or by the hyperedge \( E^k_{ij,m} \) (if \( k > \ell \)), or we have \( w = w_N \).

In the latter case, we know that \( F \) is labeled by the conjunction of the following variables:

- \( p^R \) if \( \text{vars}(\varphi) \subseteq \lambda(N_i) \) and \( \lambda_\varphi(N_i) \in \text{vars}(\varphi) \) implies \( w[g] = \varepsilon \)

- \( p^R_z \) if \( \text{vars}(\varphi) = \{ z \} \subseteq \lambda(N_i) \), \( z = \lambda_\varphi(N_i) \), and \( w[g] = R_w \)

- \( p^R_z \) and \( p^R_{z'} \) if \( \text{vars}(\varphi) = \{ z, z' \} \subseteq \lambda(N_i) \), \( z = \lambda_\varphi(N_i) \), \( z' = \lambda_\varphi'(N_i) \), and either \( w[g] = R_{w'} \) or \( w[g'] = R_{w'} \).

Since \( F \) evaluates to false, one of these variables must be assigned 0 under \( v \).

First suppose that \( p^R \) is in the label and \( v(p^R) = 0 \). Then since Equation (3) is satisfied, it must be the case that \( \varphi \) belongs to some \( q_i \), but the fact that \( \lambda_\varphi(N_i) \in \text{vars}(\varphi) \) implies \( w_N[g] = \varepsilon \) means that all variables in \( \varphi \) must belong to \( t \), contradicting the fact that \( q_i \) contains only atoms that have at least one variable in \( t \).

Next suppose that one of \( p^R_z \), \( p^R_z \), and \( p^R_{z'} \) is present because of a unary atom (item above). Then we know that there is some atom \( \varphi \) with \( \text{vars}(\varphi) = \{ z, z' \} \subseteq \lambda(N_i) \), \( z = \lambda_\varphi(N_i) \), \( z' = \lambda_\varphi'(N_i) \), and either \( w[g] = R_{w'} \) or \( w[g'] = R_{w'} \).

It follows that there is a tree witness \( t \in \Theta \) such that \( z \in t \). This means that the atom \( p^R_{z} \) must be a conjunct of Equation (3), and so it must be satisfied under \( v \).

Moreover, the fact that \( w[g] = R_{w'} \) or \( w[g'] = R_{w'} \) means that \( t_1 = R_t \), so the variables \( p^R_z \) and \( p^R_{z'} \) are also satisfied under \( v \), contradicting our earlier assumption to the contrary.

We have thus shown that \( E' \) is independent and covers all zeros under \( v \), which means that \( f_P(v) = 1 \).

For the other direction, suppose that \( f_P(v) = 1 \), i.e. there is an independent subset \( E' \) of the hyperedges in \( H_{q,T} \) that covers all vertices that evaluate to 0 under \( v \). It is clear from the construction of \( H_{q,T} \) that the set \( E' \) must contain exactly one hyperedge of the form \( E^k_i \) for every node \( N_i \) in \( T \), and exactly one hyperedge of the form \( E^k_{ij,m} \) for every edge \( \{ N_i, N_j \} \) in \( T \).

Moreover, if we have hyperedges \( E^k_i \) and \( E^k_{ij,m} \), then it must be the case that \( k = k' \).

We can thus associate to every node \( N_i \) the tuple \( w_{N_i} = \xi_k \).

Since all zeros are covered, we know that for every node \( N_i \), the following variables are assigned to 1 by \( v \):

- \( p^R \) if \( \text{vars}(\varphi) \subseteq \lambda(N_i) \) and \( \lambda_\varphi(N_i) \in \text{vars}(\varphi) \) implies \( w[g] = \varepsilon \)

- \( p^R_z \) if \( \text{vars}(\varphi) = \{ z \} \subseteq \lambda(N_i) \), \( z = \lambda_\varphi(N_i) \), and \( w[g] = R_w' \)

- \( p^R_z \) and \( p^R_{z'} \) if \( \text{vars}(\varphi) = \{ z, z' \} \subseteq \lambda(N_i) \), \( z = \lambda_\varphi(N_i) \), \( z' = \lambda_\varphi'(N_i) \), and either \( w[g] = R_{w'} \) or \( w[g'] = R_{w'} \).

We know from the definition of the set of hyperedges in \( H_{q,T} \) that every \( w_{N_i} \) is consistent with \( N_i \), and for adjacent nodes \( N_i, N_j \), pairs \( w_{N_i} \) and \( w_{N_j} \) are compatible. Using the consistency and compatibility properties, and the connectedness condition of tree decompositions, we can infer that the pairs assigned to any two nodes \( N_i, N_j \) in \( T \) are compatible.

Since every variable must appear in at least one node label, it follows that we can associate a unique word \( w_z \) to every variable \( z \) in \( q \). Now let \( \equiv \) be the smallest equivalence relation on the atoms of \( q \) that satisfies the following condition:

If \( y \in \text{vars}(q) \), \( w_y \neq \varepsilon \), \( y \in q_1 \), and \( y \in q_2 \), then \( q_1 \equiv q_2 \).
Let $q_1, \ldots, q_m$ be the queries corresponding to the equivalence classes of $\equiv$. It is easy to see that the queries $q_i$ are pairwise disjoint. Moreover, if $q_i$ contains only variables $z$ with $w_z = \varepsilon$, then $q_i$ consists of a single atom. We can show that the remaining $q_i$ correspond to tree witnesses:

**Claim.** For every $q_i$ that contains a variable $y$ with $w_y \neq \varepsilon$:

1. there is a role $R_i$ such that every $w_y \neq \varepsilon$ begins by $R_i$
2. there is a homomorphism $h_i$ from $q_i$ into $C_{T_i, A_i}$, where $T_i = T \cup \{A_{R_i} \subseteq \exists R_i\}$ and $A_i = \{A_{R_i}(a)\}$ (with $A_{R_i}$ fresh)
3. there is a tree witness $t_i$ for $q_i$ and $T$ generated by $R_i$ such that $q_i = q_{t_i}$

**Proof of claim.** From the way we defined $q_i$, we know that there exists a sequence $Q_0, \ldots, Q_m$ of subsets of $q$ such that $Q_0 = \{q_0\} \subseteq q_i$ contains a variable $y_0$ with $w_{y_0} \neq \varepsilon$, $Q_m = q_i$, and for every $1 \leq \ell \leq m$, $Q_{\ell+1}$ is obtained from $Q_{\ell}$ by adding an atom $\rho \in q \setminus Q_{\ell}$ that contains a variable $y$ that appears in $Q_{\ell}$ and is such that $w_y \neq \varepsilon$. By construction, every atom in $q_i$ contains a variable $y$ with $w_y \neq \varepsilon$. Let $R_i$ be the first letter of the word $w_{y_0}$, and for every $0 \leq \ell \leq m$, let $h_{\ell}$ be the function mapping every variable $z$ in $Q_{\ell}$ to $aw_z$. Statements 1 and 2 can be shown by induction. The base case is trivial. For the induction step, suppose that at stage $\ell$, we know that every variable $y$ in $Q_{\ell}$ with $w_y \neq \varepsilon$ begins by $R_i$, and that $h_{\ell}$ is a homomorphism of $Q_{\ell}$ into the canonical model $C_{T_i, A_i}$. We let $\varrho$ be the unique atom in $Q_{\ell+1} \setminus Q_{\ell}$. Then we know that $\varrho$ contains a variable $y$ that appears in $Q_{\ell}$ and is such that $w_y \neq \varepsilon$. If $\varrho = B(y)$, then Statement 1 is immediate. For Statement 2, we let $N$ be a node in $T$ such that $\text{vars}(\varrho) \subseteq \lambda(N)$ (such a node must exist by the definition of tree decompositions), and let $j$ be such that $\lambda_j(N) = y$. We know that $w_N$ is consistent with $N$, so $w_y = w_N[j]$ must end by a role $U$ with $U \sqsubseteq_T B$, which proves Statement 2. Next consider the other case in which $\varrho$ contains a variable other than $y$. Then $\varrho$ must be a role atom of the form $\varrho = r(y, z)$ or $\varrho = r(z, y)$. We give the argument for the case where $\varrho = r(y, z)$ (the argument for $\varrho = r(z, y)$ is entirely similar). Let $N$ be a node in $T$ such that $\text{vars}(\varrho) \subseteq \lambda(N)$, and let $j, k$ be such that $\lambda_j(N) = y$ and $\lambda_k(N) = z$. We know that $w_N$ is consistent with $N$, so one of the following must hold:

- $w_N[k] = w_N[j] \cdot U$ with $U \sqsubseteq_T r$
- $w_N[j] = w_N[k] \cdot U$ with $U \sqsubseteq_T r^-$

By definition, we have $w_y = w_N[j]$ and $w_z = w_N[k]$. Since $w_y$ begins with $R_i$, it follows that the same holds for $w_z$ unless $w_z = \varepsilon$, which shows Statement 1. Moreover, we either have (i) $w_z = w_y U$ and $U \sqsubseteq_T r$, or (ii) $w_y = w_z U$ and $U \sqsubseteq_T r^-$. In both cases, it is clear from the way we defined $h_{\ell+1}$ that it is homomorphism from $Q_{\ell+1}$ to $C_{T_{\ell+1}, A_{\ell+1}}$, so Statement 2 holds.

Statement 3 now follows from Statements 1 and 2, the definition of $q_i$, and the definition of tree witnesses. (end proof of claim)

Let $\Theta$ consist of all the tree witnesses $t_i$ obtained from the preceding claim. As the $q_i$ are known to be disjoint, we have that the set $\{q_i \mid t_i \in \Theta\}$ is independent. We aim to show that $\psi$ satisfies the disjunct of $f_{H^{\psi}}$ that corresponds to $\Theta$ (cf. Equation (3)). First consider some $\varrho \in q \setminus q_{\Theta}$. Then we know that for every variable $z$ in $\varrho$, we have $w_z = \varepsilon$. Let $N$ be a node such that $\text{vars}(\varrho) \subseteq \lambda(N)$. Then we know that
\[ \lambda_q(N) \in \text{vars}(q) \text{ implies } w_N[g] = \varepsilon. \] It follows from (\ast) that \( v(p_q) = 1. \) Next consider a variable \( p_{z=z'} \) such that there is an atom \( q \in t_1 \) with \( \text{vars}(q) = \{z, z'\} \) such that \( z \neq z'. \) Then since \( q \in q_t, \) we know that either \( w_z \neq \varepsilon \) or \( w_{z'} \neq \varepsilon. \) It follows from (\ast) that \( v(p_{z=z'}) = 1. \) Finally, consider some \( p_{R_i} \) such that \( z \in t_1. \) First suppose that there is a unary atom \( B(z) \in q_t. \) Then we know that \( w_z \neq \varepsilon, \) and so by the above claim, we must have \( w_z = R_i w'. \) It follows that there is a node \( N \) in \( T \) such that \( z = \lambda_q(N) \) and \( w_N[g] = R_i w'. \) From (\ast), we can infer that \( p_{R_i}^E \) evaluates to \( 1 \) under \( v. \) The other possibility is that there exists a role atom \( q \in q_t \) such that \( \text{vars}(q) = \{z, z'\}. \) Let \( N \) be a node in \( T \) such that \( z = \lambda_q(N) \) and \( z' = \lambda_q(N). \) Since \( q_t = q_t, \) we know that either \( w_z \neq 0 \) or \( w_{z'} \neq 0. \) From the above claim, this yields \( w_N[g] = R_i w' \) or \( w_N[g'] = R_i w'. \) We can thus apply (\ast) to obtain \( v(p_{R_i}) = 1. \) To conclude, we have shown that \( v \) satisfies one of the disjuncts of \( f_H^V \), so \( f_H^V(v) = 1. \)

For the second statement of the theorem, we recall that the tree \( T \) in the tree decomposition of \( q \) has at most \( 2|q| - 1 \) 2 nodes. The number \( M \) of elements in \( t_1(q, T) \) cannot exceed \( |W_{q,T}|^4. \) It follows that \( H_{q,T} \) contains at most \( (2|q| - 1)^2 \cdot |W_{q,T}|^4 \) hyperedges of the form \( E \) and at most \( (2|q| - 1)^2 \cdot |W_{q,T}|^2 \) hyperedges of the form \( E^{hkm}. \) We thereby obtain the desired bound of \( |H_{q,T}| \leq 8 \cdot |q|^2 \cdot |W_{q,T}|^{2t}. \)

**From TreeHGP to \( f^P_{q,T} \).** Consider a tree hypergraph \( H = (V, E) \) that is based upon the tree \( T \) whose vertices are \( v_1, \ldots, v_n. \) Let \( T^+ \) be the directed tree obtained from \( T \) by fixing \( v_1 \) as the root and orienting edges away from \( v_1. \)

We wish to construct a tree-shaped query \( q_H \) and TBox \( \mathcal{T}_{\Pi} \) of depth 2 whose primitive evaluation function \( f_{q_H,\mathcal{T}_{\Pi}} \) can be used to compute \( f_H. \) The construction generalizes the one from the preceding section for linear queries. The query \( q_H \) is obtained by doubling the edges in \( T^+: \)

\[ q_H = \exists y \bigwedge_{(v_i, y_{ij}) \in T^+} (r_{ij}(y_{ij}) \land r_{ij}'(y_{ij}, y_{ij})). \]

The TBox \( \mathcal{T}_{\Pi} \) is defined as the union of \( \mathcal{T}_e \) over all hyperedges \( e \in E. \) Consider some hyperedge \( e = (v_{i_1}, \ldots, v_{i_m}) \in E, \) and suppose w.l.o.g. that \( v_{i_1} \) is the vertex in \( e \) that is highest in \( T^+ \). Then \( \mathcal{T}_e \) contains \( B_e \subseteq \exists s_e, \exists s_e^-, \subseteq \exists s_e'^-, \) and the axioms:

\[ s_e \subseteq r_{i_1,k} \quad \text{if } \{v_{i_1}, v_k\} \in e \]
\[ s_e^- \subseteq r_{j_1,i_1}^- \quad \text{if } 1 < \ell \leq n \text{ and } (v_{j_1}, v_{i_1}) \in T^+ \text{ (so, } \{v_{j_1}, v_{i_1}\} \in e) \]
\[ s_e' \subseteq r_{j_1,k}^- \quad \text{if } \{v_{j_1}, v_k\} \in e, \{v_{j_1}, v_k\} \in T^+, \text{ and } v_{j_1} \neq v_{i_1} \]
\[ s_e'^- \subseteq r_{j_1,k}^-' \quad \text{if } \{v_{j_1}, v_k\} \in e, \{v_{j_1}, v_k\} \in T^+, \text{ and } v_{j_1} \neq v_{i_1}, \text{ for any } 1 < \ell \leq m \]

Observe that both \( q_H \) and \( \mathcal{T}_{\Pi} \) are of polynomial size in \( |H|. \)

**Theorem 12.** For every tree hypergraph \( H = (V, E) \) and for all \( \alpha : V \to \{0, 1\} \) and \( \beta : E \to \{0, 1\}, \) \( f_{\Pi}(\alpha, \beta) = 1 \) iff \( f_{q_H,\mathcal{T}_{\Pi}}(\gamma) = 1 \) where \( \gamma \) is defined as follows:

\[ \gamma(B_e) = \beta(e), \gamma(r_{ij}) = \gamma(r_{ij}') = \alpha(\{v_i, v_j\}), \text{ and } \gamma(s_e) = \gamma(s_e') = 0. \]
**Proof.** Consider some $\alpha : V \rightarrow \{0, 1\}$ and $\beta : E \rightarrow \{0, 1\}$, and let $\gamma$ be as defined in the theorem statement. Define the corresponding ABox $A_\gamma$:

$$\{B_e(a) \mid \gamma(B_e) = 1\} \cup \{r_{ij}(a, a), r'_{ij}(a, a) \mid \gamma(r_{ij}) = \gamma(r'_{ij}) = 1\}$$

For the first direction, suppose that $f_H(\alpha, \beta) = 1$. Then we know that there exists $E' \subseteq \{e \in E \mid \beta(e) = 1\}$ that is independent and covers all zeros of $\alpha$. To show $f_{A_T}(\gamma) = 1$, we must show that $T_H, A_\gamma \models q_H$. Define a mapping $h$ as follows:

- $h(y_i) = as_e s'_e$ if there exists $e = \langle v_{k_1}, \ldots, v_{k_m} \rangle \in E'$ such that $\{v_i, v_j\} \in e$, and $v_i \neq v_{k_1}$ for any $1 \leq \ell \leq m$ (i.e. $v_i$ is an internal vertex of $e$). Otherwise, $h(y_i) = a$.

- $h(y_{ij}) = as_e$ if there is some $e \in E'$ such that $\{v_i, v_j\} \in e$. Otherwise, $h(y_{ij}) = a$.

To see why $h$ is well-defined, observe that since $E'$ is independent, there can be at most one hyperedge $e \in E'$ that contains $\{v_i, v_j\}$. Further note that if $as_e$ or $as_e s'_e$ is in the image of $h$, then this means that $B_e(a) \in A_\gamma$, and so by the way we defined $T_H$, we know that $as_e$ and $as_e s'_e$ both belong to the domain of $T_H$. It remains to show that $h : q_H \rightarrow T_{H,A_\gamma}$. Consider a pair of atoms $r_{ij}(y_i, y_{ij})$, $r'_{ij}(y_{ij}, y_j)$ in $q_H$. Then $(v_i, v_j) \in E'$, so either $\alpha(\{v_i, v_j\}) = 1$ or there is some $e \in E'$ such that $\{v_i, v_j\} \in e$ and $\beta(e) = 1$. In the former case, we have $\gamma(r_{ij}) = \gamma(r'_{ij}) = 1$, so $A_\gamma$ contains $r_{ij}(a, a)$ and $r'_{ij}(a, a)$. If there is no $e \in E'$ such that $\{v_i, v_j\} \in e$ and $\beta(e) = 1$, then $h(y_i) = h(y_{ij}) = h(y_j) = a$, so the atoms $r_{ij}(y_i, y_{ij})$, $r'_{ij}(y_{ij}, y_j)$ are satisfied by $h$. Now consider the alternative in which $e = \langle v_{k_1}, \ldots, v_{k_m} \rangle \in E'$ is such that $\{v_i, v_j\} \in e$ and $\beta(e) = 1$ Then we must have $B_e(a) \in A_\gamma$, so $as_e$ and $as_e s'_e$ are present in $T_{H,A_\gamma}$. There are four possibilities to consider:

- $\{v_i, v_j\} \subseteq \{v_{k_1}, \ldots, v_{k_m}\}$ (i.e. both are boundary vertices)

  We have $h(v_i) = h(v_j) = a$ and $h(v_{ij}) = as_e$, and the TBox contains $s_e \subseteq r_{ij}$ and $s'_e \subseteq r'_{ij}$.

- $v_i \in \{v_{k_1}, \ldots, v_{k_m}\} \text{ but } v_j \notin \{v_{k_1}, \ldots, v_{k_m}\}$ (i.e. $v_i$ is a boundary vertex and $v_j$ is internal)

  We have $h(v_i) = a$, $h(v_{ij}) = as_e$, and $h(v_j) = as_e s'_e$, and the TBox contains $s_e \subseteq r_{ij}$ and $s'_e \subseteq r'_{ij}$.

- $v_j \in \{v_{k_1}, \ldots, v_{k_m}\} \text{ but } v_i \notin \{v_{k_1}, \ldots, v_{k_m}\}$ (i.e. $v_j$ is a boundary vertex and $v_i$ is internal)

  Then we have $h(v_j) = a$, $h(v_{ij}) = as_e$, and $h(v_i) = as_e s'_e$, and the TBox contains $s_e \subseteq r'_{ij}$ and $s'_e \subseteq r'_{ij}$.

- $\{v_i, v_j\} \cap \{v_{k_1}, \ldots, v_{k_m}\} = \emptyset$ (i.e. both are internal vertices)

  Then we have $h(v_i) = h(v_j) = as_e s'_e$ and $h(v_{ij}) = as_e$, and the TBox contains $s_e \subseteq r_{ij}$ and $s'_e \subseteq r'_{ij}$.

In all cases, we find that $h$ satisfies the atoms $r_{ij}(y_i, y_{ij})$, $r'_{ij}(y_{ij}, y_j)$. We can thus conclude that $h : q_H \rightarrow T_{H,A_\gamma}$.

For the other direction, suppose that $f_{q_H,T_H}(\gamma) = 1$. Then we have $T_H, A_\gamma \models q_H$. We wish to show that there exists a
subset of \( \{ e \in E \mid \beta(e) = 1 \} \) that is independent and covers all zeros of \( \alpha \). Let us define \( E' \) as the set of all \( e \in E \) such that \( h^{-1}(a_{s_e}) \neq \emptyset \) (that is, \( a_{s_e} \) is in the image of \( h \)). Note that \( E' \subseteq \{ e \in E \mid \beta(e) = 1 \} \) since the element \( a_{s_e} \) only exists if \( B_e(a) \in A_{\gamma} \), and the latter holds just in the case that \( \beta(e) = 1 \). To show that \( E' \) is independent, we establish the following claim:

**Claim.** If \( h^{-1}(a_{s_e}) \neq \emptyset \), \( y_{ij} \in \text{vars}(q_H) \), and \( \{ v_i, v_j \} \in E \), then \( h(y_{ij}) = a_{s_e} \).

**Proof of claim.** Suppose that \( h^{-1}(a_{s_e}) \neq \emptyset \), where \( e = \langle v_{k_1}, \ldots, v_{k_m} \rangle \in E' \). We may assume w.l.o.g. that \( v_{k_1} \) is the highest vertex in \( e \) according to \( T' \). Now pick some variable \( z \in h^{-1}(a_{s_e}) \) such that there is no \( z' \in h^{-1}(a_{s_e}) \) that is higher than \( z \) in \( q_H \) (here we use the ordering of variables in \( q_H \) induced by the tree \( T' \)). We first note that \( z \) cannot be of the form \( y_j \), since then there is an atom in \( q_H \) of the form \( r_{pj}(y_j, y_{j'}) \) or \( r'_{pj}(y_{ij}, y_{j}) \), and \( a_{s_e} \) does not have any outgoing \( r_{j'} \) or \( r'_{j''} \) arcs. It follows that \( z = y_{j'} \). By again considering the available arcs leaving \( a_{s_e} \), we can further see that \( \{ v_j, v_{j'} \} \in e \). We next wish to show that \( j = k_1 \). Suppose that this is not the case. Then there is some edge \( \{ v_p, v_j \} \in e \) such that \( (v_p, v_j) \in T' \). A simple examination of the axioms in \( T_H \) shows that the only way for \( h \) to satisfy the atom \( r_{pj}(y_j, y_{j'}) \) is to map \( y_j \) to \( a_{s_e} \). It follows that to satisfy that atom \( r'_{pj}(y_{j'}, y_{j}) \), we must have \( h(y_{j'}) = a_{s_e} \). This contradicts our earlier assumption that \( z = y_{j'} \) was a highest vertex in \( h^{-1}(a_{s_e}) \).

We thus have \( j = k_1 \). Now using a simple inductive argument on the distance from \( y_{k_1} \), and considering the possible ways of mapping the atoms of \( q_H \), we can show that \( h(y_{ij}) = a_{s_e} \) for every \( \{ v_i, v_j \} \in E \). (end proof of claim)

Suppose that there are two distinct hyperedges \( e, e' \in E' \) that have a non-empty intersection: \( \{ v_i, v_j \} \in e \cap e' \). We know that either \( y_{ij} \) or \( y_{j} \) belongs to \( \text{vars}(q_H) \), and we can suppose w.l.o.g. that it is the former. We can thus apply the preceding claim to obtain \( h(y_{ij}) = a_{s_e} = a_{s_{e'}} \), a contradiction. We have thus shown that \( E' \) is independent, and so it only remains to show it covers all zeros. To this end, let \( \{ v_i, v_j \} \) be such that \( \alpha(\{ v_i, v_j \}) = 0 \) and again suppose w.l.o.g. that \( y_{ij} \in \text{vars}(q_H) \). Then \( A_{\gamma} \) does not contain \( r_{ij}(a, a) \), so the only way \( h \) can satisfy the query atom \( r_{ij}(y_i, y_{ij}) \) is by mapping \( y_{ij} \) to some element \( a_{s_e} \) such that \( \{ v_i, v_j \} \in e \). It follows that there is some \( e \in E' \) such that \( \{ v_i, v_j \} \in e \), so all zeros of \( \alpha \) are covered by \( E' \). We have thus shown that \( E' \) is an independent subset of \( \{ e \in E \mid \beta(e) = 1 \} \) that covers all zeros of \( \alpha \), and hence we conclude that \( f_H(\alpha, \beta) = 1 \).

### 5.2 Tree hypergraph programs and \( \text{SAC}^1 \)

To characterize the power of tree hypergraph programs, we consider semi-unbounded fan-in circuits in which NOT gates are applied only to the inputs, AND gates have fan-in 2, and OR gates have unbounded fan-in. The complexity class \( \text{SAC}^1 \) [27] is defined by considering circuits of this type having polynomial size and logarithmic depth; \( \text{SAC}^1 \) is the non-uniform analog of the class \( \text{LOGCFL} \) of all languages logspace-reducible to context-free languages [26].

We consider semi-unbounded fan-in circuits of size \( \sigma \) and depth \( \log \sigma \), where \( \sigma \) is a parameter, and show that they are polynomially equivalent to TreeHGP by providing reductions in both directions.
Theorem 13. There exist polynomial functions $p$ and $p'$ such that:

- Every semi-unbounded fan-in circuit of size at most $\sigma$ and depth at most $\log \sigma$ is computable by a TreeHGP of size $p(\sigma)$.
- Every TreeHGP of size $\sigma$ is computable by semi-unbounded fan-in circuit of size at most $p'(\sigma)$ and depth at most $\log p'(\sigma)$.

Proof of the First Statement: TreeHGPs can simulate SAC$^1$ circuits

Consider a semi-unbounded fan-in circuit $C$ of size at most $\sigma$ and depth at most $\log \sigma$. Denote its gates by $g_1, \ldots, g_\sigma$, where $g_\sigma$ is the output gate. We define the AND-depth of gates of $C$ inductively. For the AND gate $g_i$, its AND-depth, $d(g_i)$, equals 1 if on each path from $g_i$ to the input there are no AND gates. If there are AND gates on the paths from $g_i$ to the input, consider one such gate $g_j$ with maximal AND-depth, and let $d(g_i) = d(g_j) + 1$. For an OR gate $g_i$, we let $d(g_i)$ to be equal to the largest AND-depth of an AND gate on some path from $g_i$ to the input. If there are no such AND gates, then the AND-depth of $g_i$ is 0.

We denote by $S_i$ the set of AND gates of the circuit of AND-depth $i$. Note that since the depth of $C$ is at most $\log \sigma$, we have that the AND-depth of its gates is also at most $\log \sigma$. For each AND gate $g_i$ of the circuit, we distinguish its first and its second input. We denote by Left$(g_i)$ the subcircuit computing the first input of $g_i$, that is, the subcircuit consisting of the left input $g_j$ of $g_i$ and of all gates such that there is a path from them to $g_j$. Analogously, we use Right$(g_i)$ to denote the subcircuit that computes the second input of $g_i$.

Lemma 3. Any semi-unbounded fan-in circuit $C$ of size $\sigma$ and depth $d$ is equivalent to semi-unbounded fan-in circuit of size $2^d \sigma$ and depth $d$ such that for each $i$

\[
\left( \bigcup_{g \in S_i} \text{Left}(g) \right) \cap \left( \bigcup_{g \in S_i} \text{Right}(g) \right) = \emptyset.
\]

The proof of this lemma is standard, but we include it for the sake of completeness.

Proof. We show by induction on $j$ that we can reconstruct the circuit in such a way that the property holds for all $i \leq j$, the depth of the circuit does not change and the size of the circuit increases at most by the factor of $2^j$.

Consider all AND gates in $S_j$, and consider a subcircuit $\bigcup_{g \in S_j} \text{Left}(g)$. Construct a copy $C'$ of this subcircuit separately and feed its output as first inputs to AND gates in $S_j$. This at most doubles the size of the circuit and ensures the property for $S_j$. Now for both circuits $C'$ and $\bigcup_{g \in S_j} \text{Right}(g)$ apply the induction hypothesis (note that the circuits do not intersect). The size of both circuits will increase at most by the factor of $2^{j-1}$ and the property for $S_i$ for $i < j$ will be ensured.

We can thus assume without loss of generality that the circuit $C$ satisfies the property from the preceding lemma.

We now proceed to the construction of the TreeHGP. We will begin by constructing a tree $T$ and then afterwards define a hypergraph program based upon this tree. For each
gate \( g_i \) in \( C \), we introduce three vertices \( u_i, v_i, w_i \), and we arrange these vertices into the tree from top to bottom in the following way. First we arrange vertices corresponding to the gates of depth \( k \) into a path. Vertices are ordered according to the order of gates in \( C \). In each triple of vertices, the \( u \)-vertex precedes the \( v \)-vertex, which in turn precedes the \( w \)-vertex. Next we branch the tree into two branches and associate subcircuit \( \bigcup_{g \in S_k} \text{Left}(g) \) to the left branch and the subcircuit of all other vertices to the right branch. We repeat the process for each subcircuit. This results in a tree, the number of vertices of which is \( 3\sigma \). We remove from this tree a vertex \( w_{\sigma} \).

We now define a hypergraph program based upon this tree. As before, the vertices of the hypergraph are the edges of the tree, and the hyperedges will take the form of generalized intervals. For each \( i \neq \sigma \), we introduce a hyperedge \( \{ u_i, w_i \} \). For each gate \( g_i \) if \( g_i = g_j \land g_k \), then we add a hyperedge \( \langle v_j, v_k, v_i \rangle \). If \( g_i = g_{k_1} \lor \ldots \lor g_{k_l} \), we add hyperedges \( \langle v_{k_1}, v_i \rangle, \ldots, \langle v_{k_l}, v_i \rangle \).

For input gates, we label the corresponding \( \{ u, v \} \)-edges by the corresponding literals (recall that in the circuit \( C \) negations are applied only to the inputs, so in this construction, we assume that the inputs are variables and their negations, and there are no NOT gates in the circuit). We label all other \( \{ u, v \} \)-edges and \( \{ v, w \} \)-edges of the tree by 0, and all remaining edges are labelled by 1.

The preceding construction clearly yields a TreeHGP of size polynomial in the original circuit. To complete the proof of the first statement of Theorem 13, we must show that the constructed TreeHGP computes the same function as the circuit. This is established by the following claim:

**Claim.** For a given input \( x \) and for any \( i \), the gate \( g_i \) outputs 1 iff the subtree with the root \( v_i \) can be covered.

**Proof.** We prove the claim by induction on \( i \). For input gates, the claim is trivial. If \( g_i \) is an AND gate, then both its inputs output 1. We cover both subtrees corresponding to the inputs (by induction hypothesis) and add a hyperedge \( \langle v_j, v_k, v_i \rangle \). This covers the subtree rooted in \( v_i \). If \( g_i \) is an OR gate, then there is its input \( g_k \), which is true. By induction hypothesis we cover its subtree and add a hyperedge \( \langle u_k, v_i \rangle \). All other edges of the subtree rooted in \( v_i \) can be covered by hyperedges of the form \( \{ u_p, w_p \} \).

**Proof of the Second Statement: SAC^1 circuits can simulate TreeHGPs**

Now we proceed to the second part of Theorem 13. Suppose \( H \) is a TreeHGP of size \( \sigma \), and denote by \( T \) its underlying tree. We are going to construct a semi-unbounded fan-in circuit of size polynomial in \( \sigma \). We first describe the idea of the construction, then do some preliminary work and after that actually construct a circuit.

But first of all, we note that it is not convenient to think about covering all zero vertices of the hypergraph, and it is more convenient to think about partitioning the set all vertices into disjoint hyperedges. To switch to this setting, for each vertex \( e \) of the hypergraph (recall it is an edge of \( T \)), we introduce a hyperedge \( \{ e \} \). The presence of this hyperedge will be given to a circuit by an input: the edges labeled by 1 on this particular input are presented. So, after the input is fixed, we have a hypergraph with a certain set of edges, and we have to compute whether the hypergraph has an partition into disjoint hyperedges.
Before we proceed, we need to introduce some notation related to the trees. A vertex of a tree $T$ is called a branching point if it has degree at least 3. A branch of the tree $T$ is a simple path between two branching points which does not contain any other branching points. If $v_1, v_2$ are vertices of $T$ we denote by $T_{v_1,v_2}$ the subtree of $T$ lying between the vertices $v_1$ and $v_2$. If $v$ is a vertex of degree $k$ with adjacent edges $e_1, \ldots, e_k$ then it splits $T$ into $k$ vertex-disjoint subtrees which we denote by $T_{v,e_1}, \ldots, T_{v,e_k}$. We call a vertex of a subtree $T_1$ by boundary point if it has a neighbor in $T$ outside of $T_1$. The edges of $T_1$ adjacent to boundary points are called boundary edges of $T_1$. The degree of a subtree $T_1$ is the number of its boundary points. Note that there is only one subtree of $T$ of degree 0—the tree $T$ itself.

Before we proceed with the proof of the theorem, we show the following technical lemma concerning the structure of tree hypergraph programs.

**Lemma 4.** For any tree hypergraph program $H$ with underlying tree $T$, there is an equivalent tree hypergraph program $H'$ with underlying tree $T'$ such that each hyperedge of $H'$ covers at most one branching point of $T'$, and the size of $H'$ is at most $p(|H|)$ for some explicit polynomial $p$.

**Proof (Sketch).** Let $h_1, \ldots, h_l$ be the hyperedges of $H$ containing more than 2 branching points. Let $b_{h_1}, \ldots, b_{h_l}$ be the number of branching points in them. We prove the lemma by induction on $b = \sum_i b_{h_i}$. The base case is when this sum is 0.

For the induction step, consider $h_1$ and let $v$ be one of the branching points of $T$ in $h_1$. Denote by $e_1, \ldots, e_k$ the edges adjacent to $v$ in $T$. On each $e_i$ near vertex $v$, introduce two new adjacent edges $e_{i1}, e_{i2}$, edge $e_{i1}$ closer to $v$, and label them by 0. Let $v_i$ be the new vertex lying between $e_{i1}$ and $e_{i2}$. Break $h$ into $k + 1$ several hyperedges by vertices $v_i$ and substitute $h$ by these new hyperedges. Add hyperedges $\{e_{i1}, e_{i2}\}$ for all $i$. It is not hard to see that for each evaluation of variables there is a cover of all zeros in the original hypergraph if and there is a cover of all zeros in the new hypergraph. It is also not hard to see that $b_{h}$ has decreased during this operation and the size of the hypergraph program has increased by at most $2|T|$. So the lemma follows. \(\Box\)

Thus, in what follows, we can assume that each hyperedge of the hypergraph contains at most one branching point.

Now we are ready to proceed with the proof of the theorem. The main idea of the computation generalizes the polynomial size logarithmic depth semi-unbounded fan-in circuit for directed connectivity problem (discussed below).

In what follows, we say that some subtree $T'$ of $T$ can be partitioned into disjoint hyperedges if there is a set of disjoint hyperedges $h_1, h_2, \ldots, h_k$ in $H$ such that they all lie in $T'$ and their union contains all edges of $T'$. Given the tree $H$ underlying the hypergraph, we say that its vertices $v_1$ and $v_2$ are reachable from each other if the subtree lying between them can be partitioned into disjoint hyperedges. In this case, we let $\text{Reach}(v_1, v_2) = 1$, otherwise we let $\text{Reach}(v_1, v_2) = 0$. If $v$ is a vertex of $T$ and $e = \{v, u\}$ is an edge adjacent to it, we say that $v$ is reachable from the side of $e$ if the subtree $T_{v,e}$ can be partitioned into disjoint hyperedges. In this case, we let $\text{Reach}(v, e) = 1$, otherwise we let $\text{Reach}(v, e) = 0$. Our circuit will gradually compute the reachability relation Reach for more and more vertices, and in the end, we will compute whether the whole tree can be partitioned into hyperedges.
First, our circuit will compute reachability relation for vertices on each branch of the tree $T$. If one of the endpoints of the branch is a leaf, we compute the reachability for each vertex from the side containing the leaf. This is done just like for the usual reachability problem.

Next we proceed to compute reachability between vertices on different branches of $T$. For this, consider a tree $D$ those vertices are branching points and leaves of the original tree $T$ and those edges are branches of $T$. In $D$, each vertex is either a leaf or a branching point. We will consider subtrees of the tree $D$.

We describe a process of partitioning $D$ into subtrees. At the end of the process, all subtrees will be individual edges. We have the following basic operation. Assume that we have already constructed a subtree $D'$, consider some vertex $v \in D'$ and assume that it has $k$ outgoing edges $e_1, \ldots, e_k$ within $D'$, $e_i = \{v, v_i\}$. By partitioning $D'$ in the vertex $v$, we call a substitution of $D'$ by a set of disjoint subtrees $D_1, \ldots, D_k \subseteq D'$, where for all $i$, we let $D_i = D_{v,e_i} \cap D'$.

The following lemma helps us to apply our basic operation efficiently.

**Lemma 5.** Consider a subtree $D'$ of size $m$. If its degree is $\leq 1$, then there is $v \in D'$ partitioning it into subtrees of size at most $m/2 + 1$ and degree at most 2 each. If the degree of $D'$ is 2, then there is $v \in D'$ partitioning it into subtrees of size at most $m/2 + 1$ and degree at most 2 and possibly one subtree of size less than $m$ and degree 1.

**Proof.** If $D'$ is of degree $\leq 1$, then consider its arbitrary vertex $v_1$ and subtrees into which this vertex divides $D'$. If among them there is a subtree $D_1$ larger than $m/2 + 1$, then consider the (unique) vertex $v_2$ in this subtree adjacent to $v_1$. If we separate $D'$ by $v_2$, then this partition will consist of the tree $D' \setminus D_1 \cup \{v_1, v_2\}$ and other trees lying inside of $D_1$. Thus, $D_1$ will be of size at most $m/2$ and other subtrees will be of size smaller than $|D_1|$. Thus, the size of the largest subtree decreased, and we repeat the process until the size of the largest subtree becomes at most $m/2 + 1$.

If $D'$ has degree 2, consider its boundary points $b_1$ and $b_2$. Repeat the same process starting with $v_1 = b_1$. Once in this process the current vertex $v$ tries to leave a path between $b_1$ and $b_2$, we stop. For this vertex $v$, it is not hard to see that all the resulting trees are of degree at most 2, and the only tree having the size larger than $m/2$ is of degree 1.

With this lemma, the partition process works as follows. We start with the partition $\{D\}$ consisting of the tree $D$ itself and repeatedly partition current set of subtrees into smaller one. At each step, we repeat the described procedure for each subtree separately. Note that after two steps the size of the largest subset decreases by the factor of 2. Thus in $O(\log \sigma)$ steps, we obtain the partition consisting of individual edges.

Now we are ready to describe the computational process of the circuit. The circuit will consider tree partitions described above in the reversed order. That is, we first have subtrees consisting of individual edges. Then on each step we merge some of them. In the end, we obtain the whole tree.

The intuition is that along with the construction of a subtree $D_1$, we compute the reachability for its boundary edges, that is, for example if the boundary edges of $D_1$ are $b_1$ and $b_2$ then we compute the reachability relation $\text{Reach}(v_1, v_2)$ for all $v_1$ lying on the branch $b_1$ in $T$ and $v_2$ lying on the branch $b_2$ in $T$. 
Now we are ready to describe the circuit. First for each branch of the tree it computes the reachability matrix. This is done by squaring the adjacency matrix $O((\log \sigma))^2$ times for all branches in parallel. Note that squaring a matrix requires only bounded fan-in AND-gates, and thus this step is readily computable by semi-unbounded circuit of size polynomial in $\sigma$ and depth logarithmic in $\sigma$.

Thus the circuit computes the reachability matrix for the initial partition of $D$. Next the circuit computes the reachability matrix for larger subtrees of $D$ following the process above. More specifically, suppose we merge subtrees $D_1, \ldots, D_k$ meeting in the vertex $u$ to obtain a tree $D'$. For simplicity of notation, assume that there are two subtrees $D_1$ and $D_2$ having degree 2, denote by $b, b'$ the boundary edges of $D'$ and by $b_1, \ldots, b_k$ the boundary edges of $D_1, \ldots, D_k$ respectively adjacent to the vertex $u$.

It is not hard to see that for all vertices $v$ in $b$ and $v'$ in $b'$, it is true that
\[
\text{Reach}(v, v') = \bigvee_{h \ni u} \left( \text{Reach}(v, v_1) \wedge \text{Reach}(v', v_2) \wedge \bigwedge_{i=3}^k \text{Reach}(v_i, e_i) \right),
\]
where $h$ ranges over all hyperedges of our hypergraph, $v_1, \ldots, v_k$ are boundary edges of $h$ lying in the branches $b_1, \ldots, b_k$ respectively, for each $i$, $e_i$ is an edge adjacent to $v_i$ and not contained in $h$.

The case when only one subtree among $D_1, \ldots, D_k$ has degree 2 is analogous.

Thus we have described the circuit. It is easy to see that its size is bounded by some fixed polynomial in $\sigma$. It is only left to show that this circuit can be arranged in such a way that it has depth $O(\log \sigma)$. This is not trivial since we have to show how to compute big AND in the formula above to make depth logarithmic. For this, we will use the following lemma.

**Lemma 6.** Suppose the reachability relation for each branch is already computed. Then the subtree $D'$ with $m$ edges constructed on step $i$ can be computed in the AND-depth at most $\log(m + i)$.

**Proof.** The proof proceeds by induction. Suppose that to construct $D'$, we unite subtrees $D_1, \ldots, D_k$ of sizes $m_1, \ldots, m_k$ respectively. Note that $m = m_1 + \ldots + m_k$.

By the induction hypothesis, we can compute each subtree $D_j$ within the AND-depth at most $\log m_j + i - 1$.

Consider a $k$-letter alphabet $A = \{a_1, \ldots, a_k\}$ and assign to each letter $a_j$ the probability $m_j/m$. It is well-known that there is a prefix binary code for this alphabet such that each letter $a_j$ is encoded by a word of length $\lceil \log(m/m_j) \rceil$. This encoding can be represented by a rooted binary tree the leaves of which are labeled by letters of $A$ and the length of the path from root to the leaf labeled by $a_j$ is equal to $\lceil \log(m/m_j) \rceil$. Assigning the AND function to each vertex of the tree we obtain the computation of AND in the formula above. The depth of this computation is maximum over $j$ of
\[
\log m_j + (i - 1) + \lceil \log(m/m_j) \rceil \leq \log m_j + (i - 1) + \log(m/m_j) + 1 = \log m + i.
\]

From this lemma and the fact that the computation stops after $O(\log \sigma)$ steps, we obtain that overall the AND-depth of the circuit is $O(\log \sigma)$ and thus the overall depth is $O(\log \sigma)$.
5.3 Size of rewritings of bounded treewidth queries

Theorems 11 and 13 together show us how to construct a polysize monotone $\text{SAC}^1$ circuit that computes $f'_{Hq}$. Thus, by applying Theorem 2, we obtain:

**Theorem 14.** Fix a constant $t > 0$, and let $\mathcal{I}$ be a class of OWL 2 QL TBoxes with the polynomial image property. Then all CQs of treewidth at most $t$ and TBoxes in $\mathcal{I}$ have polynomial-size NDL-rewritings.

**Proof.** Fix a constant $t > 0$, and a class $\mathcal{I}$ of OWL 2 QL TBoxes with the polynomial image property. Recall that this means that there is a polynomial $p$ such that for every TBox $T \in \mathcal{I}$ and every CQ $q$, $|W_{q,T}| \leq p(|q| + |T|)$. By using this property, the fact that $t$ is a fixed constant, and Theorem 11, we have that there is a polynomial $p'$ such that for any CQ $q$ of treewidth at most $t$ and any Tbox $T \in \mathcal{I}$ the TreeHGP $H_{q,T}$ computes $f'_{Hq}$ and is of size at most $p'(|q| + |T|)$. Now we apply Theorem 13 and conclude that $f'_{Hq}$ may be computed by a semi-unbounded fan-in circuit of size $p''(p'(|q| + |T|))$ for some polynomial $p''$. By Theorem 2, there exist NDL rewritings for $q$ and $T$ with this upper bound for the size. \hfill $\square$

In the case of FO-rewritings, we can show that the existence of polysize rewritings corresponds to the open question of whether $\text{SAC}^1 \subseteq \text{NC}^1$.

**Theorem 15.** The following are equivalent:

1. There exist polysize FO-rewritings for all tree-shaped CQs and depth 2 TBoxes;
2. There exist polysize FO-rewritings for all CQs of treewidth at most $t$ and TBoxes from a class $\mathcal{I}$ with the polynomial image property (for any fixed $t$);
3. There exists a polynomial function $p$ such that every semi-unbounded fan-in circuit of size at most $\sigma$ and depth at most $\log \sigma$ is computable by a formula of size $p(\sigma)$.

Equivalently, $\text{SAC}^1 \subseteq \text{NC}^1$.

**Proof.**

$(2) \implies (1)$: Trivial.

$(1) \implies (3)$: Suppose (1) holds. In other words, there exists a polynomial $p''$ such that every tree-shaped query $q$ and TBox $T$ of depth 2 has a rewriting of the size $p''(|q| + |T|)$. Consider a semi-unbounded fan-in circuit $C$ of size $\sigma$ and depth at most $\log \sigma$ that computes the Boolean function $f$. By Theorem 13, it gives rise to a Boolean function which is computed by a TreeHGP $P$ based on a tree hypergraph $H$ of size at most $p(\sigma)$ for the polynomial $p$ from Theorem 13. By Theorem 12, there exists a tree-shaped query $q_H$ and a TBox of depth 2 $T_H$ such that $f_P$ is a subfunction of $f'_{q_H}$. By the assumption, there exists a FO rewriting for $q_H$ and $T_H$ of size at most $p''(|q| + |T|)$. This number is polynomial in $\sigma$ (take the composition of $p$, the polynomial function from Theorem 12 and $p''$). Now by Theorem 3, there exists a polysize first-order formula for computing $f'_{q_H}$, and hence also for $f_P$ and $f$.

$(3) \implies (2)$: Suppose (3) holds. Fix $t$. Take query $q$ of treewidth at most $t$ and a TBox $T$. Since $t$ fixed, by Theorem 11, there is a polysize TreeHGP $P$ that computes $f'_{Hq}$. By assumption and Theorem 13, there is a polysize FO-formula computing $f'_{Hq}$. We can then apply Theorem 2 to transform it into an FO-rewriting of $q$ and $T$. \hfill $\square$
To complete the picture, we generalize the result of [15] that says that all tree-shaped queries have poly-size PE-rewritings with respect to ontologies of depth 1, by showing that this property applies also to the wider class of bounded treewidth queries.

**Theorem 16.** Fix a constant \( t > 0 \). Then there exist polysize PE-rewritings for all CQs of treewidth at most \( t \) and TBoxes of depth 1.

**Proof.** Fix a constant \( t > 0 \). Throughout the proof, we will consider CQs of treewidth at most \( t \), and for every such query \( q \), we will use \( \text{mtd}_t(q) \) (for ‘minimal tree decomposition’) to denote the minimum number of vertices over all tree decomposition of \( q \) that have width at most \( t \).

We also fix a TBox \( T \) of depth 1, and as before, we use \( \Theta_T^q \) to denote the set of all tree witnesses for the query \( q \) and TBox \( T \). Since \( T \) has depth 1, it is known from [15] that every tree witness \( t = (t_i, t_i) \in \Theta_T^q \) contains a unique interior point (i.e. \( |t_i| = 1 \)), and no two distinct tree witnesses may share the same interior point.

Given a set \( U \subseteq \text{vars}(q) \) of variables, we will use \( \Theta_T^q(U) \) to refer to the set of all tree witnesses whose interior point belongs to \( U \). If \( \Omega \) is an independent subset of \( \Theta_T^q(U) \), then the set \( \text{bvars}(U, \Omega) \) of border variables for \( U \) and \( \Omega \) is defined as follows:

\[
\{ u \in U \mid \text{there is no } t \in \Omega \text{ with } t_i = \{ u \} \} \cup \{ z \mid z \in t_i \text{ for some } t_i \in \Omega \}.
\]

We also define \( q^{U,\Omega} = q \setminus \{ \varphi \in q \mid \text{vars}(\varphi) \subseteq U \cup \text{bvars}(U, \Omega) \} \). For \( z, z' \in \text{vars}(q) \setminus (V \cup \text{bvars}(U, \Omega)) \), we set \( z \sim z' \) if there is a path in \( G_q \) from \( z \) to \( z' \) that does not pass through \( \text{bvars}(U, \Omega) \) (recall that \( G_q \) is undirected). The \( \sim \) relation can be lifted to the atoms in \( q^{U,\Omega} \) by setting \( (q_1 \cup q_2) \sim q_1 \) if all of the variables in \( (q_1 \cup q_2) \setminus \text{bvars}(U, \Omega) \) are \( \sim \)-equivalent. Let \( \Theta_T^{q_i^{U,\Omega}} \) denote the queries formed by the \( \sim \)-equivalence classes of atoms in \( q^{U,\Omega} \) with \( \text{avars}(q_i^{U,\Omega}) = (\text{avars}(q_i) \cup \text{bvars}(U, \Omega)) \cap \text{vars}(q_i^{U,\Omega}) \).

**Claim.** For every query \( q \) of treewidth \( t \), there exists a subset \( U \) of \( \text{vars}(q) \) such that \( |U| \leq t + 1 \) and for all subqueries \( q_i^{U,\Omega} \), we have \( \text{mtd}_t(q_i^{U,\Omega}) < \text{mtd}_t(q)/2 \).

**Proof of claim.** Consider some tree decomposition \((T, \lambda)\) of \( q \) of width \( t \) with \( T = (V, E) \) and \( |V| = \text{mtd}_t(q) \). It was shown in [15] that there exists a vertex \( v \in V \) such that each connected component in the graph \( T_v \) obtained by removing \( v \) from \( T \) has at most \( |V|/2 = \text{mtd}_t(q)/2 \) vertices. Consider the set of variables \( U = \lambda(v) \). Since \( (T, \lambda) \) has width \( t \), we have that \( |U| \leq t + 1 \). As to the second property, we observe that it is sufficient to consider queries of the form \( q_i^{U,\emptyset} \), since by definition, \( q_i^{U,\emptyset} \subseteq q_i^{U,\emptyset} \) for any \( \Omega \subseteq \Theta_T^q(U) \). We then remark that due to the connectedness condition on tree decompositions, and the fact that we only consider paths in \( G_q \) that do not pass by variables in \( U = \lambda(v) \), every query \( q_i^{U,\emptyset} \) can be obtained by:

1. taking a connected component \( C_i = (V_i, E_i) \) in the graph \( T_v \);
2. considering the resulting tree decomposition \((C_i, \lambda_i)\), where \( \lambda_i \) is the restriction of \( \lambda \) to the vertices in \( C_i \);
3. taking a subset of the atoms in \( \{ \varphi \in q \mid \text{vars}(\varphi) \subseteq \lambda(v') \text{ for some } v' \in V_i \} \).

It then suffices to recall that each connected component contains at most \( \text{mtd}_t(q)/2 \) vertices. (end proof of claim)
We now use the claim to define a recursive rewriting procedure. Given a query \( q \) of treewidth at most \( t \), we choose a set \( U \) of variables that satisfies the properties of preceding claim, and define the rewriting \( q^\dagger \) of \( q \) w.r.t. \( T \) as follows:

\[
q^\dagger = \exists y \bigg( \bigwedge_{\Omega \subseteq \Theta'^*} (\text{at}(U, \Omega) \land \text{tw}(U, \Omega) \land \bigwedge_{i=1}^{k} (q_{U,\Omega}^i)^\dagger) \bigg)
\]

where

- \( y \) is the set of all existential variables in \( U \),
- \( \text{at}(U, \Omega) = \{ \varrho \in q \mid \text{vars}(\varrho) \subseteq U \text{ and there is no } t \in \Omega \text{ with } \varrho \in q_{T}^t \} \);
- \( \text{tw}(U, \Omega) = \bigwedge_{t \in \Omega} \left( \exists z' \left( \bigvee_{t \in \Theta} q_{T}^t[\mathcal{R}] \land \bigwedge_{z \in \mathcal{R}} (z = z') \right) \right) \);
- \( (q_{U,\Omega}^i)^\dagger \) are rewritings of the queries \( q_{U,\Omega}^i \), which are constructed recursively according to the same procedure.

The \( \dagger \)-rewriting we have just presented generalizes the rewriting procedure for tree-shaped queries from [15], and correctness can be shown similarly to the original procedure. As to the size of the obtained rewriting, we remark that since the set \( U \) is always chosen according to the claim, and \( \text{mtd}_t(q) \leq (2|q| - 1)^2 \) (cf. [16]), we know that the depth of the recursion is logarithmic in \( |q| \). We also know that the branching is at most \( 2t+1 \) at each step, since there is at most one recursive call for each subset \( \Omega \subseteq \Theta'^* \), and since \( T \) has depth 1, we have \( \Theta'^*(U) \leq |U| \). Thus, the resulting formula \( q^\dagger \) has the structure of a tree whose number of nodes is bounded by \( O((2t+1)^{\log |q|}) = O(|q|^{t+1}) \).

6 Conclusion and Future Work

In this paper, we filled some gaps in the succinctness landscape for query rewriting in OWL 2 QL by providing new bounds on the worst-case rewriting size for various forms of tree-like queries. In doing so, we closed one of the open questions from [15].

In future work, we plan to consider additional dimensions of the succinctness landscape. For example, all existing lower bounds rely on sequences \( (q_n, T_n) \) in which the number of roles in \( q_n \) and \( T_n \) grows with \( n \). Moreover, \( T_n \) often contains a large number of role inclusions. Thus, an interesting and practically relevant problem is to explore the impact of restricting the number of roles and/or the use of role inclusions.

References

1. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69, 2009.

3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

4. M. Bienvenu, M. Ortiz, M. Simkus, and G. Xiao. Tractable queries for lightweight description logics. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*. AAAI Press, 2013.

5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

6. A. Chortaras, D. Trivela, and G. Stamou. Optimized query rewriting for OWL 2 QL. In *Proc. of the 23rd Int. Conf. on Automated Deduction (CADE-23)*, volume 6803 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2011.

7. T. Eiter, M. Ortiz, M. Šimkus, T.-K. Tran, and G. Xiao. Query rewriting for Horn-SHIQ plus rules. In *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012)*. AAAI Press, 2012.

8. G. Gottlob, S. Kikot, R. Kontchakov, V. Podolskii, T. Schwentick, and M. Zakharyaschev. The price of query rewriting in ontology-based data access. *Artificial Intelligence*, 213:42–59, 2014.

9. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In V. Vianu and C. H. Papadimitriou, editors, *Proc. of the 18th ACM SIGACFSIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1999)*, pages 21–32. ACM Press, 1999.

10. G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2012)*, pages 254–263. AAAI Press, 2012.

11. S. Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer, 2012.

12. M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc.of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 539–550. ACM Press, 1988.

13. S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. Exponential lower bounds and separation for query rewriting. In *Proc. of the 39th Int. Colloquium on Automata, Languages, and Programming (ICALP 2012)*, Part II, volume 7392 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 2012.

14. S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. On the succinctness of query rewriting over OWL 2 QL ontologies with shallow chases. In *Proc. of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014)*. ACM Press, 2014.

15. T. Kloks. *Treewidth: Computation and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

16. M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo. A sound and complete backward chaining algorithm for existential rules. In *Proc. of the 6th Int. Conf. on Web Reasoning and Rule Systems (RR 2012)*, volume 7497 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2012.

17. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language profiles. W3C Recommendation, 11 December 2012. Available at http://www.w3.org/TR/owl2-profiles/.
19. M. Ortiz and M. Simkus. Reasoning and query answering in description logics. In *Reasoning Web*, volume 7487 of *Lecture Notes in Computer Science*, pages 1–53. Springer, 2012.
20. H. Pérez-Urbina, B. Motik, and I. Horrocks. A comparison of query rewriting techniques for DL-Lite. In *Proc. of the 22nd Int. Workshop on Description Logics (DL 2009)*, volume 477. CEUR-WS, 2009.
21. H. Pérez-Urbina, E. Rodríguez-Díaz, M. Grove, G. Konstantinidis, and E. Sirin. Evaluation of query rewriting approaches for OWL 2. In *Proc. of SSWS+HPCSW 2012*, volume 943. CEUR-WS, 2012.
22. A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *Proc. of the 8th Int. Symposium on Fundamentals of Computation Theory (FCT’91)*, volume 529 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 1991.
23. M. Rodríguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013)*, volume 8218 of *Lecture Notes in Computer Science*, pages 558–573. Springer, 2013.
24. R. Rosati. Prexto: Query rewriting under extensional constraints in DL-Lite. In *Proc. of the 9th Extended Semantic Web Conf. (EWSC 2012)*, volume 7295 of *Lecture Notes in Computer Science*, pages 360–374. Springer, 2012.
25. R. Rosati and A. Almatelli. Improving query answering over DL-Lite ontologies. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pages 290–300. AAAI Press, 2010.
26. H. Venkateswaran. Properties that characterize LOGCFL. *J. Computer and System Sciences*, 43(2):380–404, 1991.
27. H. Vollmer. *Introduction to circuit complexity - a uniform approach*. Texts in theoretical computer science. Springer, 1999.
28. A. Wigderson. *The complexity of graph connectivity*. Springer, 1992.
29. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases (VLDB’81)*, pages 82–94. IEEE Computer Society, 1981.