Query Rewriting over Shallow Ontologies

S. Kikot[1], R. Kontchakov[1], V. Podolskii[2], and M. Zakharyaschev[1]

[1] Department of Computer Science and Information Systems
Birkbeck, University of London, U.K.
{kikot,roman,michael}@dcs.bbk.ac.uk
[2] Steklov Mathematical Institute, Moscow, Russia
podolskii@mi.ras.ru

Abstract. We investigate the size of conjunctive query rewritings over *OWL 2 QL* ontologies of depth 1 and 2 by means of a new formalism, called hypergraph programs, for computing Boolean functions. Both positive and negative results are obtained. All conjunctive queries over ontologies of depth 1 have polynomial-size nonrecursive datalog rewritings; tree-shaped queries have polynomial-size positive existential rewritings; however, for some queries and ontologies of depth 1, positive existential rewritings can only be of superpolynomial size. Both positive existential and nonrecursive datalog rewritings of conjunctive queries and ontologies of depth 2 suffer an exponential blowup in the worst case, while first-order rewritings can grow superpolynomially unless NP $\subseteq$ P/poly.

1 Introduction

This paper is a continuation of the series [14, 12, 13], where we investigated the following problems. Let $\boldsymbol{q}(\boldsymbol{x})$ be a conjunctive query (CQ) with answer variables $\boldsymbol{x}$ and let $\mathcal{T}$ be an *OWL 2 QL* ontology. It is known (see, e.g., [8, 3]) that there exists a first-order formula $\boldsymbol{q}'(\boldsymbol{x})$, called an FO-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$, such that $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\boldsymbol{a})$, for any ABox $\mathcal{A}$ and any vector $\boldsymbol{a}$ of individuals in the ABox (of the same length as $\boldsymbol{x}$). Thus, to find certain answers to $\boldsymbol{q}(\boldsymbol{x})$ over $\mathcal{T}$ and $\mathcal{A}$, it suffices to find answers to $\boldsymbol{q}'(\boldsymbol{x})$ over the data $\mathcal{A}$, which can (hopefully) be done by conventional relational database management systems (RDBMSs). Various experiments showed, however, that rewritings $\boldsymbol{q}'$ can be too large for the RDBMSs to cope with. This put forward the followings problems:

- What is the overhead of answering CQs via ontologies compared to standard database query answering in the worst case?
- What is the size of FO-rewritings of CQs and *OWL 2 QL* ontologies in the worst case?
- Can rewritings of one type (say, nonrecursive datalog) be substantially shorter that rewritings of another type (say, positive existential)?
- Are there interesting and useful sufficient conditions on CQs and ontologies under which rewritings are short?

We showed [14, 12, 13] that, for a certain sequence of (tree-shaped) CQs $\boldsymbol{q}_n$ and *OWL 2 QL* TBoxes $\mathcal{T}_n$, the problem '$\mathcal{A} \models \boldsymbol{q}_n$?' is in P for combined complexity, while the problem '$(\mathcal{T}_n, \mathcal{A}) \models \boldsymbol{q}_n$?' is NP-complete. Moreover, any positive existential (PE) or nonrecursive datalog (NDL) rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of exponential size, while any FO-rewriting is of superpolynomial size unless NP $\subseteq$ P/poly. We also showed that NDL-rewritings are in general exponentially more succinct than PE-rewritings, and FO-rewritings can be superpolynomially more succinct than PE-rewritings. On the other hand, Gottlob and Schwentick [9] demonstrated that one can always find a polynomial-zise rewriting for the price of polynomially-many additional existential quantifiers over a domain with at least two constants (thus confirming once again that formalisms with nondeterminism are exponentially more succinct; cf. also [5]). Finally, Kikot *et el.* [15] give a practically useful sufficient condition on CQs and ontologies under which PE-rewritings are of polynomial size.

The problem we address in this paper is whether the depth of TBoxes (that is, the maximal depth of the canonical models with single-individual ABoxes) has any impact on the size of rewritings. (The TBoxes $\mathcal{T}_n$ mentioned above are of depth n.) In particular, what happens if we restrict the depth of TBoxes to 1 or 2? (PE-rewritings over TBoxes of depth 0 are trivially polynomial.) The obtained results are summarised below:

(1) For any CQ and TBox of depth 1, there is a polynomial-size NDL-rewriting.
(2) PE-rewritings of some CQs and TBoxes of depth 1 are of superpolynomial size.
(3) All tree-shaped CQs and TBoxes of depth 1 have a polynomial-size PE-rewriting.
(4) For TBoxes of depth 2, both NDL- and PE-rewritings can suffer an exponential blowup, while FO-rewritings can suffer a superpolynomial blowup (unless NP $\subseteq$ P/poly).

Moreover, it follows from our constructions that the problem of finding short FO-rewritings for given CQ and a TBox (of depth 2) is equivalent to the problem of finding short Boolean circuits for NP-complete problems.

We begin by observing that the tree-witness PE-rewritings, representing all possible homomorphisms of subqueries of a given CQ to the canonical models with one ABox individual, give rise to a class of monotone Boolean functions associated with hypergraphs and called hypergraph functions. In particular, hypergraphs H of degree 2 (every vertex in which belongs to at most 2 hyperedges) correspond to Boolean CQs $\boldsymbol{q}_H$ and TBoxes $\mathcal{T}_H$ of depth 1 such that answering $\boldsymbol{q}_H$ over $\mathcal{T}_H$ and single-individual ABoxes amounts to computing the hypergraph function for H. We show then that representing Boolean functions as hypergraphs of degree 2 is polynomially equivalent to representing them by nondeterministic branching programs (NBPs) [11]. This correspondence and known results about NBPs [18, 10] give **(1)** and **(2)** above. We show **(3)** using the tree form of CQs and the fact that, over TBoxes of depth 1, CQs $\boldsymbol{q}$ can only have $\leq |\boldsymbol{q}|$ tree witnesses. To obtain **(4)**, we observe that hypergraphs of degree > 2 are computationally as powerful as nondeterministic Boolean circuits (NP/poly) and encode computing the function $\textsc{Clique}_{n,k}(\boldsymbol{e})$ (a graph with n vertices has a k-clique) as answering some CQs over TBoxes of depth 2 (which correspond to hypergraphs of degree 3). Although hypergraph programs for Boolean functions are introduced as a technical means to investigate the size of CQ rewritings, they may be of independent interest to the complexity theory of Boolean functions.

2 *OWL 2 QL* and Rewritings over H-complete ABoxes

In this paper, we use the following (simplified) syntax of *OWL 2 QL*. It contains *individual names* a_i, *concept names* A_i, and *role names* P_i ($i \geq 1$). *Roles* R and *basic concepts* B are defined by the grammar:

$$R \quad ::= \quad P_i \quad \mid \quad P_i^-, \qquad\qquad B \quad ::= \quad \bot \quad \mid \quad A_i \quad \mid \quad \exists R.$$

A *TBox*, $\mathcal{T}$, is a finite set of *inclusions* of the form

$$B_1 \sqsubseteq B_2, \qquad B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqsubseteq R_2, \qquad R_1 \sqcap R_2 \sqsubseteq \bot.$$

An *ABox*, $\mathcal{A}$, is a finite set of atoms of the form $A_k(a_i)$ or $P_k(a_i, a_j)$. The set of individual names in $\mathcal{A}$ is denoted by $\mathsf{ind}(\mathcal{A})$. $\mathcal{T}$ and $\mathcal{A}$ together form the *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The semantics for *OWL 2 QL* is defined in the usual way based on interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ [6]. We write $B_1 \equiv B_2$ as a shortcut for $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq B_1$.

For every role name R in $\mathcal{T}$, we take two fresh concept names A_R, A_{R^-} and add to $\mathcal{T}$ the axioms $A_R \equiv \exists R$ and $A_{R^-} \equiv \exists R^-$. We say that the resulting TBox is in *normal form* and assume, without loss of generality, that every TBox in this paper is in normal form. We denote by $\sqsubseteq_{\mathcal{T}}$ the subsumption relation induced by $\mathcal{T}$ and write $S_1 \sqsubseteq_{\mathcal{T}} S_2$ if $\mathcal{T} \models S_1 \sqsubseteq S_2$, where S_1, S_2 are both concepts or roles. We say that an ABox $\mathcal{A}$ is *H-complete with respect to* $\mathcal{T}$ in case

$$\begin{aligned} R_2(a,b) \in \mathcal{A} &\quad \text{if} \quad R_1(a,b) \in \mathcal{A} \text{ and } R_1 \sqsubseteq_{\mathcal{T}} R_2,\\ A_2(a) \in \mathcal{A} &\quad \text{if} \quad A_1(a) \in \mathcal{A} \text{ and } A_1 \sqsubseteq_{\mathcal{T}} A_2, \end{aligned}$$

for all concept names A_i (including the A_R) and roles R_i. We write $R(a,b) \in \mathcal{A}$ for $P(a,b) \in \mathcal{A}$ if $R = P$ and for $P(b,a)$ if $R = P^-$; also, we write $A_R(a) \in \mathcal{A}$ if $R(a,b) \in \mathcal{A}$, for some b.

A *conjunctive query* (CQ) $\boldsymbol{q}(\boldsymbol{x})$ is a formula $\exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, where φ is a conjunction of atoms of the form $A_k(z_1)$ or $P_k(z_1, z_2)$ with $z_i \in \boldsymbol{x} \cup \boldsymbol{y}$ (without loss of generality, we assume that CQs do not contain constants). A tuple $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$ is a *certain answer* to $\boldsymbol{q}(\boldsymbol{x})$ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I} \models \boldsymbol{q}(\boldsymbol{a})$ for all $\mathcal{I} \models \mathcal{K}$; in this case we write $\mathcal{K} \models \boldsymbol{q}(\boldsymbol{a})$. If $\boldsymbol{x} = \emptyset$, the CQ $\boldsymbol{q}$ is called *Boolean*; a certain answer to such a $\boldsymbol{q}$ over $\mathcal{K}$ is 'yes' if $\mathcal{K} \models \boldsymbol{q}$ and 'no' otherwise. Where convenient, we regard a CQ $\boldsymbol{q}$ as the set of its atoms.

Suppose $\mathcal{T}$ is a TBox and $\boldsymbol{q}(\boldsymbol{x})$ a CQ. An FO-formula $\boldsymbol{q}'(\boldsymbol{x})$ with free variables $\boldsymbol{x}$ and without constants is an *FO-rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$ *over H-complete ABoxes* if, for any H-complete (with respect to $\mathcal{T}$) ABox $\mathcal{A}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}'(\boldsymbol{a})$. If an FO-rewriting $\boldsymbol{q}'$ is a positive existential formula, we call it a *PE-rewriting* of $\boldsymbol{q}$ and $\mathcal{T}$. We also consider rewritings in the form of nonrecursive Datalog queries. We remind the reader that a *Datalog program*, Π, is a finite set of Horn clauses $\forall \boldsymbol{x}\, (\gamma_1 \wedge \cdots \wedge \gamma_m \to \gamma_0)$, where each γ_i is an atom of the form $P(x_1, \ldots, x_l)$ with $x_i \in \boldsymbol{x}$. The atom γ_0 is called the *head* of the clause, and $\gamma_1, \ldots, \gamma_m$ its *body*. All variables occurring in the head must also occur in the body. A predicate P *depends* on a predicate Q in Π if Π contains a clause whose head is P and whose body contains Q. Π is called *nonrecursive* if this dependence relation for Π is acyclic. For a nonrecursive Datalog program Π and an atom $\boldsymbol{q}'(\boldsymbol{x})$, we say that $(\Pi, \boldsymbol{q}')$ is an *NDL-rewriting of* $\boldsymbol{q}(\boldsymbol{x})$ *and* $\mathcal{T}$ *over H-complete ABoxes* in case $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\Pi, \mathcal{A} \models \boldsymbol{q}'(\boldsymbol{a})$, for any H-complete ABox $\mathcal{A}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$. Rewritings *over arbitrary ABoxes* are defined by dropping the condition that the ABoxes are H-complete.

Proposition 1. *Suppose $(\Pi, \boldsymbol{q}')$ is an NDL-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes. Then there is an NDL-rewriting $(\Pi', \boldsymbol{q}')$ of $\boldsymbol{q}$ and $\mathcal{T}$ over arbitrary ABoxes such that $|\Pi'| \leq |\Pi| + O(|\mathcal{T}|^2)$. A similar result holds for PE- and FO-rewritings.*

Proof. We assume without loss of generality that $\boldsymbol{q}'$ is not a concept or role name in $\mathcal{T}$. Let Π^* be the result of replacing each A and P in Π with fresh predicates A^* and P^*, where A is a concept and P a role name, respectively. Define Π' to be the union of Π^* and the following clauses:

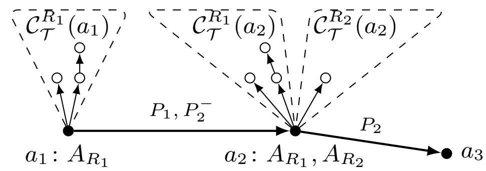
$$A^*(x) \leftarrow B(x), \qquad \text{for concept names } A \text{ and concepts } B \text{ with } B \sqsubseteq_{\mathcal{T}} A,$$
$$P^*(x,y) \leftarrow R(x,y), \qquad \text{for role names } P \text{ and roles } R \text{ with } R \sqsubseteq_{\mathcal{T}} P,$$

where $B(x) = R(x,z)$, for a fresh z, if $B = \exists R$ and $R(x,y) = S(x,y)$ if $R = S$ and $R(x,y) = S(y,x)$ if $R = S^-$, for a role name S. It should be clear that $(\Pi', \boldsymbol{q}')$ is an NDL-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over arbitrary ABoxes. The cases of PE- and FO-rewritings are similar (except that in these cases we replace each A and P with a disjunction of the bodies in their defining clauses). ❑

3 The Tree-Witness Rewriting

In this section, we define one particular PE-rewriting over H-complete ABoxes, which will be used to establish links with formulas and circuits computing certain monotone Boolean functions.

Recall [8, 15] that, for any TBox $\mathcal{T}$ and ABox $\mathcal{A}$, there is a *canonical model* $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$ such that $(\mathcal{T}, \mathcal{A}) \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$, for all CQs $\boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$. The domain of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ consists of the individuals in $\mathsf{ind}(\mathcal{A})$ and the witnesses introduced by the existential quantifiers in $\mathcal{T}$. Every individual $a \in \mathsf{ind}(\mathcal{A})$ with $(\mathcal{T}, \mathcal{A}) \models A_R(a)$ is a root of a (possibly infinite) sub-tree $\mathcal{C}^R_{\mathcal{T}}(a)$ of $\mathcal{C}_{\mathcal{T},\mathcal{A}}$, which may intersect another such tree only on their common root a. Every $\mathcal{C}^R_{\mathcal{T}}(a)$ is isomorphic to the canonical model of $(\mathcal{T}, \{A_R(a)\})$.



We say that $\mathcal{T}$ is *of depth* ω if at least one of $\mathcal{C}_{\mathcal{T}}^{R}(a)$ is infinite; $\mathcal{T}$ *is of depth* d, $1 \le d < \omega$, if there is a chain of the form $w_0 R_0 w_1 \dots w_{d-1} R_{d-1} w_d$ in the trees $\mathcal{C}_{\mathcal{T}}^{R}(a)$, R a tole in $\mathcal{T}$, but there is no such chain of greater length.

By definition, $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ iff there is a homomorphism $h\colon \boldsymbol{q}(\boldsymbol{a}) \to \mathcal{C}_{\mathcal{T},\mathcal{A}}$. Such a homomorphism h splits $\boldsymbol{q}$ into the subquery mapped by h to $\mathsf{ind}(\mathcal{A})$ and the subquery mapped to the trees $\mathcal{C}_{\mathcal{T}}^{R}(a)$. We can think of a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ as listing possible splits of $\boldsymbol{q}$ into such subqueries.

Suppose $\boldsymbol{q}' \subseteq \boldsymbol{q}$ and there is a homomorphism $h\colon \boldsymbol{q}' \to \mathcal{C}_{\mathcal{T}}^{R}(a)$, for some a, such that h maps all answer variables in $\boldsymbol{q}'$ to a. Let $\mathsf{t_r} = h^{-1}(a)$ and let $\mathsf{t_i}$ be the remaining set of (existentially quantified) variables in $\boldsymbol{q}'$. Suppose $\mathsf{t_i} \neq \emptyset$. We call the pair $\mathsf{t} = (\mathsf{t_r}, \mathsf{t_i})$ a *tree witness for* $\boldsymbol{q}$ *and* $\mathcal{T}$ *generated by* R if the query $\boldsymbol{q}'$ is a *minimal* subset of $\boldsymbol{q}$ such that, for any $y \in \mathsf{t_i}$, every atom in $\boldsymbol{q}$ containing y belongs to $\boldsymbol{q}'$. In this case, we denote $\boldsymbol{q}'$ by $\boldsymbol{q}_\mathsf{t}$. By definition, we have

$$\boldsymbol{q}_\mathsf{t} = \big\{ S(\boldsymbol{z}) \in \boldsymbol{q} \mid \boldsymbol{z} \subseteq \mathsf{t_r} \cup \mathsf{t_i} \text{ and } \boldsymbol{z} \not\subseteq \mathsf{t_r} \big\}.$$

Note that the same tree witness $\mathsf{t} = (\mathsf{t_r}, \mathsf{t_i})$ can be generated by different roles R. We denote the set of all such roles by Ω_t and define the formula

$$\mathsf{tw_t} \quad = \quad \bigvee_{R \in \Omega_\mathsf{t}} \exists z \,\big(A_R(z) \quad \land \quad \bigwedge_{x \in \mathsf{t_r}} (x = z) \big). \tag{1}$$

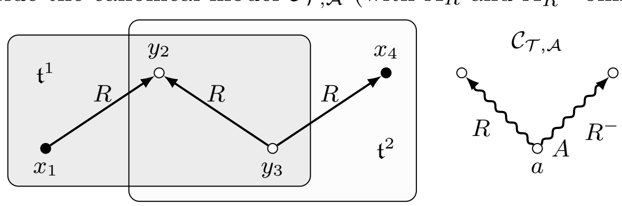
Tree witnesses t and t' are *consistent* if $\boldsymbol{q}_\mathsf{t} \cap \boldsymbol{q}_{\mathsf{t}'} = \emptyset$. Each *consistent set* Θ of tree witnesses (in which any pair of distinct tree witnesses is consistent) determines a subquery $\boldsymbol{q}_\Theta$ of $\boldsymbol{q}$ that comprises all atoms of $\boldsymbol{q}_\mathsf{t}$, for $\mathsf{t} \in \Theta$. The subquery $\boldsymbol{q}_\Theta$ is to be mapped to the $\mathcal{C}_{\mathcal{T}}^{R}(a)$, whereas the remainder, $\boldsymbol{q} \setminus \boldsymbol{q}_\Theta$, obtained by removing the atoms of $\boldsymbol{q}_\Theta$ from $\boldsymbol{q}$, is mapped to $\mathsf{ind}(\mathcal{A})$. The following PE-formula $\boldsymbol{q}_\mathsf{tw}$ is called the *tree-witness rewriting* of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes:

$$\boldsymbol{q}_\mathsf{tw}(\boldsymbol{x}) \quad = \quad \bigvee_{\Theta \text{ consistent}} \exists \boldsymbol{y} \,\Big((\boldsymbol{q} \setminus \boldsymbol{q}_\Theta) \;\land\; \bigwedge_{\mathsf{t} \in \Theta} \mathsf{tw_t} \Big). \tag{2}$$

Example 1. Consider the KB $\mathcal{K} = (\mathcal{T}, \{A(a)\})$, where

$$\mathcal{T} = \{A \sqsubseteq \exists R,\; A \sqsubseteq \exists R^-,\; A_R \equiv \exists R,\; A_{R^-} \equiv \exists R^-\},$$

and the CQ $\boldsymbol{q}(x_1, x_4) = \{R(x_1, y_2), R(y_3, y_2), R(y_3, x_4)\}$ shown in the picture below alongside the canonical model $\mathcal{C}_{\mathcal{T},\mathcal{A}}$ (with A_R and A_{R^-} omitted).



There are two tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$: $\mathfrak{t}^1 = (\mathfrak{t}^1_\mathsf{r}, \mathfrak{t}^1_\mathsf{i})$ generated by R and $\mathfrak{t}^2 = (\mathfrak{t}^2_\mathsf{r}, \mathfrak{t}^2_\mathsf{i})$ generated by R^-, with

$$\mathfrak{t}^1_\mathsf{r} = \{x_1, y_3\}, \qquad \mathfrak{t}^1_\mathsf{i} = \{y_2\}, \qquad \mathsf{tw}_{\mathfrak{t}^1} = \exists z\,(A_R(z) \land (x_1 = z) \land (y_3 = z)),$$
$$\mathfrak{t}^2_\mathsf{r} = \{y_2, x_4\}, \qquad \mathfrak{t}^2_\mathsf{i} = \{y_3\}, \qquad \mathsf{tw}_{\mathfrak{t}^2} = \exists z\,(A_{R^-}(z) \land (x_4 = z) \land (y_2 = z)).$$

We have $\boldsymbol{q}_{\mathfrak{t}^1} = \{R(x_1, y_2), R(y_3, y_2)\}$ and $\boldsymbol{q}_{\mathfrak{t}^2} = \{R(y_3, y_2), R(y_3, x_4)\}$, so $\mathfrak{t}^1$ and $\mathfrak{t}^2$ are inconsistent. Thus, we obtain the following tree-witness rewriting:

$$\boldsymbol{q}_{\mathsf{tw}}(x_1, x_4) \;=\; \exists y_2, y_3\big[(R(x_1, y_2) \land R(y_3, y_2) \land R(y_3, x_4)) \lor (R(y_3, x_4) \land \mathsf{tw}_{\mathfrak{t}^1}) \lor (R(x_1, y_2) \land \mathsf{tw}_{\mathfrak{t}^2})\big].$$

Theorem 1. *For any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}_{\mathsf{tw}}(\boldsymbol{a})$.*

Note that $|\boldsymbol{q}_{\mathsf{tw}}| = O(|\varXi| \cdot |\boldsymbol{q}| \cdot |\mathcal{T}|)$, where $\varXi$ is the collection of all consistent sets of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ and the $|\mathcal{T}|$ factor comes from the $\mathsf{tw}_\mathfrak{t}$-formulas and multiple roles that may generate a tree witness. Note also that the number of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ may be exponential in $\boldsymbol{q}$ [15]. If any two tree-witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ are *compatible*, that is, they are either consistent or one is included in the other, then $\boldsymbol{q}_{\mathsf{tw}}$ can be equivalently transformed into the PE-rewriting

$$\boldsymbol{q}'_{\mathsf{tw}}(\boldsymbol{x}) \;=\; \exists \boldsymbol{y} \bigwedge_{S(\boldsymbol{z}) \in \boldsymbol{q}} \Big(S(\boldsymbol{z}) \;\lor \bigvee_{\substack{\mathfrak{t} \text{ a tree witness for } \boldsymbol{q} \text{ and } \mathcal{T} \\ S(\boldsymbol{z}) \in \boldsymbol{q}_\mathfrak{t}}} \mathsf{tw}_\mathfrak{t} \Big),$$

which is of polynomial size whenever the number of tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ is polynomial. Our aim now is to investigate transformations of this kind in the more abstract setting of Boolean functions. In Section 8, we shall see an example of $\boldsymbol{q}$ and $\mathcal{T}$ with only $|\boldsymbol{q}|$-many tree witnesses any PE-rewriting of which is of superpolynomial size because of multiple combinations of inconsistent tree witnesses.

4 Hypergraph Functions

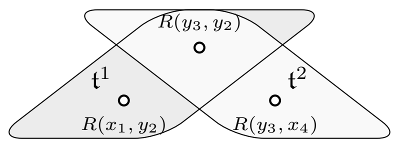
The tree-witness rewriting $\boldsymbol{q}_{\mathsf{tw}}$ above gives rise to monotone Boolean functions we call hypergraph functions. For the complexity theory of monotone Boolean functions, the reader is referred to [2, 11].

Let $H = (V, E)$ be a hypergraph with *vertices* $v \in V$ and *hyperedges* $e \in E$, $E \subseteq 2^V$. We call a subset $X \subseteq E$ *independent* if $e \cap e' = \emptyset$, for any distinct $e, e' \in X$. The set of vertices that occur in the hyperedges of X is denoted by V_X. 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

$$f_H \;=\; \bigvee_{X \text{ independent}} \Big(\bigwedge_{v \in V \setminus V_X} p_v \;\land\; \bigwedge_{e \in X} p_e \Big). \tag{3}$$

The rewriting $\boldsymbol{q}_{\mathsf{tw}}$ of $\boldsymbol{q}$ and $\mathcal{T}$ defines a hypergraph whose vertices are the atoms of $\boldsymbol{q}$ and hyperedges are the sets $\boldsymbol{q}_{\mathfrak{t}}$, for tree witnesses $\mathfrak{t}$ for $\boldsymbol{q}$ and $\mathcal{T}$. We denote this hypergraph by $H^{\boldsymbol{q}}_{\mathcal{T}}$. The formula (3) defining $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is basically the same as the rewriting (2) with the atoms $S(\boldsymbol{z}) \in \boldsymbol{q}$ and tree witness formulas $\mathsf{tw}_{\mathfrak{t}}$ treated as propositional variables. We denote these variables by $p_{S(\boldsymbol{z})}$ and $p_{\mathfrak{t}}$ (rather than p_v and p_e), respectively.

Example 2. Consider again the CQ $\boldsymbol{q}$ and TBox $\mathcal{T}$ from Example 1. The hypergraph $H^{\boldsymbol{q}}_{\mathcal{T}}$ is shown in the picture below



and

$$f_{H^{\boldsymbol{q}}_{\mathcal{T}}} = (p_{R(x_1,y_2)} \land p_{R(y_3,y_2)} \land p_{R(y_3,x_4)}) \lor (p_{R(y_3,x_4)} \land p_{\mathfrak{t}^1}) \lor (p_{R(x_1,y_2)} \land p_{\mathfrak{t}^2}).$$

Suppose the function $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is computed by some Boolean formula $\chi_{H^{\boldsymbol{q}}_{\mathcal{T}}}$. Consider the FO-formula $\widehat{\chi}_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ obtained by replacing each $p_{S(\boldsymbol{z})}$ in $\chi_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ with $S(\boldsymbol{z})$, each $p_{\mathfrak{t}}$ with $\mathsf{tw}_{\mathfrak{t}}$, and adding the prefix $\exists \boldsymbol{y}$. By comparing (3) and (2), we see that the resulting FO-formula is a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes. This gives the first claim in the following theorem:

Theorem 2. *Suppose $\boldsymbol{q}$ is a CQ and $\mathcal{T}$ a TBox.*

(i) If the function $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is computed by a propositional Boolean formula $\chi_{H^{\boldsymbol{q}}_{\mathcal{T}}}$, then $\widehat{\chi}_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is an FO-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes.
(ii) If $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ is computed by a monotone Boolean circuit $\mathbf{C}$, then there is an NDL-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over H-complete ABoxes of size $O(|\mathbf{C}| \cdot (|\boldsymbol{q}| + |\mathcal{T}|))$.

Proof. We only prove (ii). First, we define a unary predicate D_0 by the rules

$$D_0(z) \leftarrow A(z), \tag{4}$$

for every concept name A (including the A_R) in $\mathcal{T}$ and $\boldsymbol{q}$. Assuming that $\boldsymbol{x}$ and $\boldsymbol{y}$ are answer and existential variables in $\boldsymbol{q}$, respectively, we then set $\boldsymbol{z} = \boldsymbol{x} \cup \boldsymbol{y}$ and define the $|\boldsymbol{z}|$-ary predicate

$$D(\boldsymbol{z}) \leftarrow \bigwedge_{z \in \boldsymbol{z}} D_0(z). \tag{5}$$

We need the predicate D to ensure that all the rules in our datalog program are safe (that is, every variable in the head occurs in the body).

Suppose $H^{\boldsymbol{q}}_{\mathcal{T}}$ has m vertices and l edges. Let $g_1, \dots, g_n$ be the nodes of $\mathbf{C}$ ordered in such a way that $g_1, \dots, g_m$ correspond to the atoms $S_1(\boldsymbol{z}_1), \dots, S_m(\boldsymbol{z}_m)$ of $\boldsymbol{q}$, $g_{m+1}, \dots, g_{m+l}$ correspond to the tree witnesses $\mathfrak{t}^1, \dots, \mathfrak{t}^l$ generated by roles in sets $\Omega_1, \dots, \Omega_l$, respectively, and $g_{m+l+1}, \dots, g_n$ correspond to the gates of $\mathbf{C}$ with g_n its output. For $1 \le i \le m$, we take the rules

$$G_i(\boldsymbol{z}) \leftarrow S_i(\boldsymbol{z}_i) \land D(\boldsymbol{z}). \tag{6}$$

For $m < i \le m + l$, take the rules

$$G_i(\boldsymbol{z}) \leftarrow A_R(z_0) \land \bigwedge_{y \in \mathfrak{t}_{\mathsf{r}}^{i-m}} (z_0 = y) \land D(\boldsymbol{z}), \quad \text{for } R \in \Omega_{i-m}. \tag{7}$$

where z_0 is a fresh variable. For $i > m + l$, we take the rules

$$G_i(\boldsymbol{z}) \leftarrow G_j(\boldsymbol{z}) \land G_k(\boldsymbol{z}) \land D(\boldsymbol{z}), \qquad \text{if } g_i = g_j \land g_k, \tag{8}$$

$$\begin{array}{l} G_i(\boldsymbol{z}) \leftarrow G_j(\boldsymbol{z}) \land D(\boldsymbol{z}), \\ G_i(\boldsymbol{z}) \leftarrow G_k(\boldsymbol{z}) \land D(\boldsymbol{z}), \end{array} \qquad \text{if } g_i = g_j \lor g_k. \tag{9}$$

Denote the resulting set of rules (4)–(9) by Π. We claim that (Π, G_n) is an NDL-rewriting of $\boldsymbol{q}$ and $\mathcal{T}$ over complete ABoxes. To see this, we can transform (Π, G_n) to a PE-formula of the form

$$\exists \boldsymbol{y} \Big[\psi(\boldsymbol{x}, \boldsymbol{y}) \land \bigwedge_{z \in \boldsymbol{x} \cup \boldsymbol{y}} \Big(\bigvee_{A \text{ in } \boldsymbol{q}, \mathcal{T}} A(z)\Big)\Big],$$

where $\exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y})$ can be constructed by taking the Boolean formula representing $\mathbf{C}$ and replacing $p_{S(\boldsymbol{z})}$ with $S(\boldsymbol{z})$ and $p_{\mathfrak{t}}$ with $\mathsf{tw}_{\mathfrak{t}}$. It follows from part (i) that $\exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y})$ is a rewriting of $\boldsymbol{q}$ and $\mathcal{T}$. It should be clear that the big conjunction does not change this fact. ❑

Thus, the problem of constructing short rewritings is reducible to the problem of finding short Boolean formulas computing the hypergraph functions. Hypergraphs of *degree* ≤ 2, in which every vertex belongs to at most two hyperedges, are of particular interest to us because (i) TBoxes of depth one have hypergraphs of degree ≤ 2, and (ii) their hypergraph functions are the functions computed by branching programs of polynomial size.

We call a hypergraph H *representable* if there are a CQ $\boldsymbol{q}$ and a TBox $\mathcal{T}$ such that H is isomorphic to $H_{\mathcal{T}}^{\boldsymbol{q}}$. A hypergraph is said to be of *degree 2* if every vertex in it belongs to *exactly* two hyperedges. In the next section, we show that hypergraphs of degree 2 are representable by means of TBoxes of depth 1, and, conversely, all CQs over TBoxes of depth 1 define hypergraphs of degree ≤ 2.

5 Hypergraphs of Degree 2 and TBoxes of Depth 1

Theorem 3. (i) *If $\boldsymbol{q}$ is a CQ and $\mathcal{T}$ a TBox of depth one, then the hypergraph $H_{\mathcal{T}}^{\boldsymbol{q}}$ is of degree ≤ 2.*

(ii) *The number of distinct tree witnesses for $\boldsymbol{q}$ and $\mathcal{T}$ does not exceed the number of variables in $\boldsymbol{q}$.*

Proof. We have to show that every atom in $\boldsymbol{q}$ belongs to at most two $\boldsymbol{q}_{\mathfrak{t}}$, $\mathfrak{t}$ a tree witness for $\boldsymbol{q}$ and $\mathcal{T}$. Suppose $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ is a tree witness (generated by some R) and $y \in \mathfrak{t}_\mathsf{i}$. Then $\mathfrak{t}_\mathsf{r}$ consists of all those variables z in $\boldsymbol{q}$ for which $S(y, z) \in \boldsymbol{q}$ or $S(z, y) \in \boldsymbol{q}$, for some S. By the definition of tree witness, $\mathfrak{t}_\mathsf{i} = \{y\}$. Thus, different tree witnesses have different 'internal' variables y. An atom of the form $A(u) \in \boldsymbol{q}$ is in $\boldsymbol{q}_{\mathfrak{t}}$ iff $u = y$. An atom of the form $P(u, v) \in \boldsymbol{q}$ is in $\boldsymbol{q}_{\mathfrak{t}}$ iff either $u = y$ or $v = y$. In other words, $P(u, v) \in \boldsymbol{q}$ can only be covered by the tree witness with internal u and by the tree witness with internal v. ❑

Let $H = (V, E)$ be a hypergraph of degree 2. We can assume that it comes with two fixed maps $i_1, i_2 \colon V \to E$ such that $i_1(v) \neq i_2(v)$, $v \in i_1(v)$ and $v \in i_2(v)$, for any $v \in V$. We now define a Boolean CQ $\boldsymbol{q}_H$ and a TBox $\mathcal{T}_H$ such that H is isomorphic to $H^{\boldsymbol{q}_H}_{\mathcal{T}_H}$. For every hyperedge $e \in E$, we take an individual variable z_e and denote by $\boldsymbol{z}$ the vector of all such variables. For every vertex $v \in V$, we take a role name R_v and set:

$$\boldsymbol{q}_H \;=\; \exists \boldsymbol{z} \bigwedge_{v \in V} R_v(z_{i_1(v)}, z_{i_2(v)}).$$

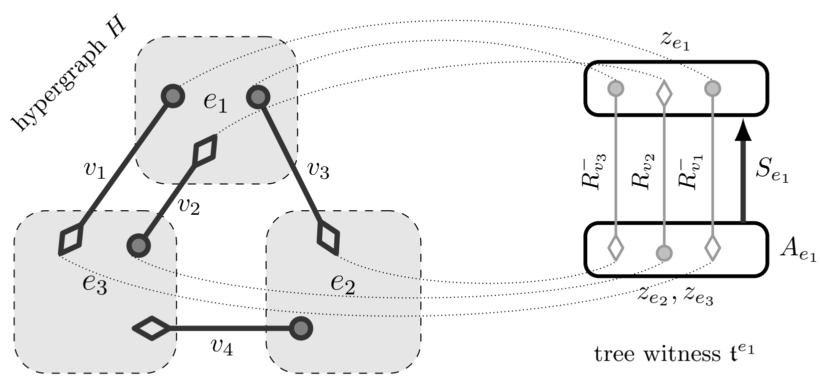
For every hyperedge $e \in E$, let A_e be a concept name and S_e a role name. Consider the TBox $\mathcal{T}_H$ with the following inclusions, for each $e \in E$:

$$\begin{aligned} &A_e \equiv \exists S_e, && \\ &S_e \sqsubseteq R_v^-, && \text{for } v \in V \text{ with } i_1(v) = e, \\ &S_e \sqsubseteq R_v, && \text{for } v \in V \text{ with } i_2(v) = e. \end{aligned}$$

Example 3. Let $H = (V, E)$ with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3\}$, where $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4\}$, $e_3 = \{v_1, v_2, v_4\}$. Suppose also that

$$\begin{aligned} i_1 &\colon v_1 \mapsto e_1, \quad v_2 \mapsto e_3, \quad v_3 \mapsto e_1, \quad v_4 \mapsto e_2, \\ i_2 &\colon v_1 \mapsto e_3, \quad v_2 \mapsto e_1, \quad v_3 \mapsto e_2, \quad v_4 \mapsto e_3. \end{aligned}$$

The hypergraph H is shown in the picture below, where each vertex v_i is represented by an edge, $i_1(v_i)$ is indicated by the circle-shaped end of the edge and $i_2(v_i)$ by the diamond-shaped end of the edge; the hyperedges e_j are shown as large grey squares:



Then

$$\boldsymbol{q}_H = \exists z_{e_1} z_{e_2} z_{e_3} \big(R_{v_1}(z_{e_1}, z_{e_3}) \land R_{v_2}(z_{e_3}, z_{e_1}) \land R_{v_3}(z_{e_1}, z_{e_2}) \land R_{v_4}(z_{e_2}, z_{e_3})\big)$$

and the TBox $\mathcal{T}_H$ contains the following inclusions:

$$\begin{array}{llll} A_{e_1} \equiv \exists S_{e_1}, & S_{e_1} \sqsubseteq R_{v_1}^-, & S_{e_1} \sqsubseteq R_{v_2}, & S_{e_1} \sqsubseteq R_{v_3}^-, \\ A_{e_2} \equiv \exists S_{e_2}, & S_{e_2} \sqsubseteq R_{v_3}, & S_{e_2} \sqsubseteq R_{v_4}^-, & \\ A_{e_3} \equiv \exists S_{e_3}, & S_{e_3} \sqsubseteq R_{v_1}, & S_{e_3} \sqsubseteq R_{v_2}^-, & S_{e_3} \sqsubseteq R_{v_4}. \end{array}$$

The canonical model $\mathcal{C}_{\mathcal{T}_H}^{S_{e_1}}(a)$ is shown on the right-hand side of the picture above. We observe now that each variable z_e uniquely determines the tree witness $\mathfrak{t}^e$ with $\boldsymbol{q}_{\mathfrak{t}^e} = \{R_v(z_{i_1(v)}, z_{i_2(v)}) \mid v \in e\}$; $\boldsymbol{q}_{\mathfrak{t}^e}$ and $\boldsymbol{q}_{\mathfrak{t}^{e'}}$ are consistent iff $e \cap e' \neq \emptyset$. It follows that H is isomorphic to $H_{\mathcal{T}_H}^{\boldsymbol{q}_H}$.

In fact, this example generalises to the following:

Theorem 4. *Any hypergraph H of degree 2 is representable; more precisely, H is isomorphic to $H_{\mathcal{T}_H}^{\boldsymbol{q}_H}$.*

Proof. We show that the map $h\colon v \mapsto R_v(z_{i_1(v)}, z_{i_2(v)})$ is an isomorphism between H and $H_{\mathcal{T}_H}^{\boldsymbol{q}_H}$. By the definition of $\boldsymbol{q}_H$, h is a bijection between V and the atoms of $\boldsymbol{q}_H$. For any $e \in E$, there is a tree witness $\mathfrak{t}^e = (\mathfrak{t}^e_\mathsf{r}, \mathfrak{t}^e_\mathsf{i})$ generated by S_e with

$$\mathfrak{t}^e_\mathsf{i} = \{z_e\} \quad \text{and } \mathfrak{t}^e_\mathsf{r} = \{z_{e'} \mid e' \cap e \neq \emptyset\},$$

and $\boldsymbol{q}_{\mathfrak{t}^e}$ consists of the $h(v)$, for $v \in e$. Conversely, every tree witness $\mathfrak{t}$ for $\boldsymbol{q}_H$ and $\mathcal{T}_H$ contains $z_e \in \mathfrak{t}_\mathsf{i}$, for some $e \in E$, and so $\boldsymbol{q}_\mathfrak{t} = \{h(v) \mid v \in e\}$. ❑

We now show that answering the CQ $\boldsymbol{q}_H$ over $\mathcal{T}_H$ and certain single-individual ABoxes amounts to computing the Boolean function f_H. Let $H = (V, E)$ be a hypergraph of degree 2 with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$. We denote by $\boldsymbol{\alpha}(v_i)$ the i-th component of $\boldsymbol{\alpha} \in \{0,1\}^n$ and by $\boldsymbol{\beta}(e_j)$ the j-th component of $\boldsymbol{\beta} \in \{0,1\}^m$. Define a single-individual ABox $\mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ by taking

$$\mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad = \quad \{R_{v_i}(a,a) \mid \boldsymbol{\alpha}(v_i) = 1\} \quad \cup \quad \{A_{e_j}(a) \mid \boldsymbol{\beta}(e_j) = 1\}.$$

Theorem 5. *Let $H = (V, E)$ be a hypergraph of degree 2. Then*

$$(\mathcal{T}_H, \mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}) \models \boldsymbol{q}_H \quad \textit{iff} \quad f_H(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1,$$

for any $\boldsymbol{\alpha} \in \{0,1\}^{|V|}$ and $\boldsymbol{\beta} \in \{0,1\}^{|E|}$.

Proof. ($\Leftarrow$) Let X be an independent subset of E such that $\bigwedge_{v \in V \setminus V_X} p_v \land \bigwedge_{e \in X} p_e$ is true on $\boldsymbol{\alpha}$ (for the p_v) and $\boldsymbol{\beta}$ (for the p_e). Define $h\colon \boldsymbol{q}_H \to \mathcal{C}_{\mathcal{T}_H, \mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}}$ by taking

$$h(z_e) = \begin{cases} w_e, & \text{if } e \in X, \\ a, & \text{otherwise,} \end{cases}$$

where w_e is the element of the canonical model $\mathcal{C}_{\mathcal{T}_H,\mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}}$ introduced to witness $\exists S_e$. It is readily checked that h is a homomorphism, and so $(\mathcal{T}_H, \mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}) \models \boldsymbol{q}_H$.

($\Rightarrow$) Suppose that $h\colon \boldsymbol{q}_H \to \mathcal{C}_{\mathcal{T}_H,\mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}}$ is a homomorphism. We show that the set $X = \{e \in E \mid h(z_e) \neq a\}$ is independent. Indeed, if $e, e' \in X$ and $v \in e \cap e'$, then h sends one end of the R_v-atom to the witness w_e and the other end to the witness $w_{e'}$, which is impossible. We claim then that $f_H(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1$. Indeed, for each $v \in V \setminus V_X$, h sends both ends of the R_v-atom to a, and so $\boldsymbol{\alpha}(v) = 1$. For each $e \in X$, since $h(z_e) \neq a$, we must have $h(z_e) = w_e$, and so $\boldsymbol{\beta}(e) = 1$. It follows that $f_H(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1$. ❑

6 Hypergraphs of Degree 2 and NBPs

In this section, we show that a Boolean function is 'computed' by a hypergraph of degree 2 iff it can be computed by a nondeterministic branching program (a switching-and-rectifier network) [11].

Let $p_1, \dots, p_n$ be propositional variables. An *input* to a hypergraph or nondeterministic branching program is a vector $\boldsymbol{\alpha} \in \{0,1\}^n$ assigning a truth value $\boldsymbol{\alpha}(p_i)$ to each variable p_i, $1 \leq i \leq n$. We extend this notation to negated variables and constants: $\boldsymbol{\alpha}(\neg p_i) = \neg\boldsymbol{\alpha}(p_i)$, $\boldsymbol{\alpha}(0) = 0$ and $\boldsymbol{\alpha}(1) = 1$.

A *hypergraph program* is a hypergraph $H = (V, E)$ with each vertex labelled by 0, 1, p_i or $\neg p_i$. We say that the hypergraph program H *computes* a Boolean function f in case, for any input $\boldsymbol{\alpha}$, we have $f(\boldsymbol{\alpha}) = 1$ iff there is an independent subset $X \subseteq E$ covering all *zeros*—the vertices with labels ℓ such that $\boldsymbol{\alpha}(\ell) = 0$. The *size* of a hypergraph program is the number of hyperedges in it. A hypergraph program is *monotone* if there are no negated variables among its vertex labels. In the remainder of this section, we concentrate on hypergraph programs H of degree ≤ 2.

It turns out that the monotone hypergraph programs capture the computational power of hypergraph functions. Note first that a monotone hypergraph program H computes the subfunction of f_H obtained by setting $p_e = 1$, for all $e \in E$, and setting p_v to be equal to the label of v. On the other hand, any hypergraph function f_H can be computed by a small hypergraph program.

Lemma 1. *For any hypergraph H of degree ≤ 2 with N hyperedges, there is a monotone hypergraph program H' of degree ≤ 2 and size $2N$ computing the function f_H.*

Proof. Consider a hypergraph $H = (V, E)$ and label each of its vertices v by a variable p_v. For each hyperedge $e \in E$, we add two fresh vertices a_e, b_e labelled by 1 and p_e, respectively; then we create a new hyperedge $e' = \{a_e, b_e\}$ and add the vertex a_e to e. We claim that the resulting hypergraph program H' computes f_H. Indeed, for any input $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha}(p_e) = 0$, we have to include the edge e' into the cover, and so cannot include the edge e itself. Thus, the program outputs 1 iff there is an independent set X of hyperedges with $\boldsymbol{\alpha}(p_e) = 1$, for all $e \in X$, covering all zeros of the variables p_v. It follows that H' computes f_H. ❑

Lemma 2. *If f is computable by a (monotone) hypergraph program of degree ≤ 2 and size N, then it can also be computed by a (monotone) hypergraph program of size $N+3$ such that all its vertices are of degree 2.*

Proof. Consider a hypergraph program of degree ≤ 2 computing f. We extend it with three vertices, x, y and z, labelled by $1, 0$ and 0, respectively, and three hyperedges $e_1 = \{v_1, \dots, v_l, x, y\}$, $e_2 = \{v_1, \dots, v_k, x, z\}$ and $e_3 = \{y, z\}$, where $v_1, \dots, v_k$ are vertices of degree 0 and $v_{k+1}, \dots, v_l$ vertices of degree 1. It is easy to see that each cover should contain e_3 but cannot contain e_1, e_2. Indeed, y and z should both be covered. However, e_1 and e_2 intersect and cannot be both in the same cover. Thus, y and z should be covered by e_3, while e_1 and e_2, intersecting e_3, are not in the cover. After these choices we are left with the original hypergraph. Clearly, this construction preserves monotonicity. ❑

A *nondeterministic branching program* (NBP) [11] is a directed multigraph with two distinguished vertices, s (*source*) and t (*sink*), and the arcs labelled by 0, 1, p_i or $\neg p_i$ (the arcs of the first type have no effect, the arcs of the second type are called *rectifiers*, and those of the third and fourth types *contacts*). We assume that s has no incoming arcs and t has no outgoing arcs, and note that NBPs may have multiple parallel arcs (with distinct labels) connecting two nodes. We say that *t is reachable from s under $\boldsymbol{\alpha}$* ($s \to_{\boldsymbol{\alpha}} t$, in symbols) if there is a directed path from s to t such that each arc of the path is labelled by ℓ with $\boldsymbol{\alpha}(\ell) = 1$. An NBP *computes* a Boolean function f if $f(\boldsymbol{\alpha}) = 1$ just in case $s \to_{\boldsymbol{\alpha}} t$. The size of an NBP is the number of arcs in it. We denote by $\mathrm{NBP}(f)$ the minimal size of NBPs computing f. An NBP is *monotone* if it has no negated variables in its labels. For a monotone f, we denote by $\mathrm{NBP}_+(f)$ the minimal size of monotone NBPs computing f.

We are going to show now that a Boolean function f is computable by a polynomial-size NBP iff $\neg f$ is computable by a polynomial-size hypergraph program of degree 2.

Lemma 3. *Any Boolean function f is computable by a hypergraph program of degree ≤ 2 and size $2\mathrm{NBP}(\neg f)$.*

Any monotone Boolean function f is computable by a monotone hypergraph program of degree ≤ 2 and size $2\mathrm{NBP}_+(f^*)$, where f^* is the Boolean function dual to f.

Proof. We only prove the first claim; the second is proved by the same argument.

Let $\neg f$ be computable by an NBP G. We construct a hypergraph program of degree ≤ 2 as follows. For each labelled arc e in G, the hypergraph has two vertices e^0 and e^1, which represent the beginning and the end of the arc. The vertex e^0 is labelled by the *negated* label of e in G and e^1 is labelled by 1. We also take a vertex t labelled by 0. For each arc e in G, the hypergraph has an *e-hyperedge* $\{e^0, e^1\}$. For each vertex v in G but s and t, the hypergraph has a *v-hyperedge* that consists of all vertices e^1, for the arcs e leading to v, and all vertices e^0, for the arcs e leaving v. For the vertex t, the hypergraph contains a hyperedge that consists of t and all vertices e^1, for the arcs e leading to t.

We claim that the constructed hypergraph program computes f. Indeed, if $s \not\to_{\boldsymbol{\alpha}} t$ in G then the following subset of hyperedges is independent and covers all zeros: all e-hyperedges, for the arcs e reachable from s and labelled by ℓ with $\alpha(\ell) = 1$, and all v-hyperedges with $s \not\to_{\boldsymbol{\alpha}} v$. Conversely, if $s \to_{\boldsymbol{\alpha}} t$ then it can be shown by induction that, for each arc e_i of the path, the e_i-hyperedge must be in the cover of all zeros. Thus, no independent set can cover t, which is labelled by 0. ❑

Lemma 4. *If f is computable by a hypergraph program of degree 2 and size N, then $\neg f$ can be computed by an NBP of size $O(N^3)$.*

If f is computable by a monotone hypergraph program of degree 2 and size N, then f^* can be computed by a monotone NBP of size $O(N^3)$.

Proof. We prove the first claim; the second is shown using the same argument.

Let H be a hypergraph program of degree 2 with hyperedges $e_1, \dots, e_N$. We first provide a graph-theoretic characterisation of independent sets covering all zeros based on the implication graph [4] (or the chain criterion of [7, Lemma 8.3.1]). Fix an input $\boldsymbol{\alpha}$ and consider a set $\Phi_{\boldsymbol{\alpha}}$ of propositional binary clauses with variables p_i, for $1 \le i \le N$, consisting of

– $\neg p_i \vee \neg p_j$ if $e_i \cap e_j \neq \emptyset$ (informally: intersecting hyperedges cannot be chosen at the same time),
– $p_i \vee p_j$ if there is $v \in e_i \cap e_j$ such that $\boldsymbol{\alpha}(v) = 0$ (informally: all zeros must be covered; note that all vertices have at most two incident edges).

By inspection of the definition, X is an independent set covering all zeros iff $X = \{e_i \mid 1 \le i \le N \text{ and } \boldsymbol{\beta}(p_i) = 1\}$, for some assignment $\boldsymbol{\beta}$ satisfying $\Phi_{\boldsymbol{\alpha}}$. By [7, Lemma 8.3.1], $\Phi_{\boldsymbol{\alpha}}$ is satisfiable iff there is no e_i with a (directed) cycle going through both e_i^+ and e_i^- in the directed graph $B_{\boldsymbol{\alpha}} = (V, E_{\boldsymbol{\alpha}})$, where

$$\begin{aligned} V &= \big\{e_i^+, e_i^- \mid 1 \le i \le N\big\}, \\ E_{\boldsymbol{\alpha}} &= \big\{(e_i^+, e_j^-) \mid e_i \cap e_j \neq \emptyset\big\} \;\cup\; \big\{(e_i^-, e_j^+) \mid v \in e_i \cap e_j \text{ with } \boldsymbol{\alpha}(v) = 0\big\}. \end{aligned}$$

(V is the set of all 'literals' for the variables of $\Phi_{\boldsymbol{\alpha}}$ and $E_{\boldsymbol{\alpha}}$ is the arcs for the implicational form of the clauses of $\Phi_{\boldsymbol{\alpha}}$; note that $\neg p_i \vee \neg p_j$ gives rise to two implications, $p_i \to \neg p_j$ and $p_j \to \neg p_i$, and so to two arcs in the graph). It will be convenient for us to regard the $B_{\boldsymbol{\alpha}}$, for assignments $\boldsymbol{\alpha}$, as a single labelled directed graph B with arcs of the from (e_i^+, e_j^-) labelled by 1 and arcs of the form (e_i^-, e_j^+) labelled by $\neg v$, for all $v \in e_i \cap e_j$. It should be clear that $B_{\boldsymbol{\alpha}}$ has a cycle going through e_i^+ and e_i^- iff e_i^+ is reachable from e_i^- and the other way round, or, using the NBP notation, iff $e_i^- \to_{\boldsymbol{\alpha}} e_i^+$ and $e_i^+ \to_{\boldsymbol{\alpha}} e_i^-$ in B. Thus, the required NBP contains two distinguished vertices, s and t, and, for each hyperedge e_i, two separate copies, B_i^+ and B_i^-, of B with arcs from s to the e_i^- vertex of B_i^+, from the e_i^+ vertex of B_i^+ to the e_i^+ vertex of B_i^- and from the e_i^- vertex of B_i^- to t. This construction guarantees that $s \to_{\boldsymbol{\alpha}} t$ iff there is e_i such that $B_{\boldsymbol{\alpha}}$ contains a cycle going through e_i^+ and e_i^-. ❑

As is well-known (see, e.g., [18]), in terms of the expressive power NBPs sit between Boolean formulas and Boolean circuits. And, as shown above, the computational power of monotone hypergraphs of degree 2 is the same as that of monotone NBPs. Thus, a Boolean function computable by some monotone Boolean formula can also be computed by an at most polynomially larger monotone NBP, and so, by Lemma 3, by an at most polynomially larger monotone hypergraph program, for some hypergraph of degree 2. On the other hand, a Boolean function computable by some monotone hypergraph program of degree 2 is computable, by Lemma 4, by an at most polynomially larger monotone NBP, and so by an at most polynomially larger monotone Boolean circuit.

In the next section, we shall see that monotone hypergraphs of unbounded degree (in fact, degree 3) are substantially more powerful than monotone hypergraphs of degree 2; more precisely, polynomial-size monotone hypergraph programs of degree 3 can compute NP-hard Boolean functions.

7 Hypergraph Programs and Nondeterministic Circuits

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is computed by a *nondeterministic* Boolean circuit $\mathbf{C}(\boldsymbol{x},\boldsymbol{y})$, for $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$, if for each $\boldsymbol{x}$ we have $f(\boldsymbol{x}) = 1$ iff there is $\boldsymbol{y}$ such that $\mathbf{C}(\boldsymbol{x},\boldsymbol{y}) = 1$. The variables in $\boldsymbol{y}$ are called *advice variables*. We say that a nondeterministic circuit $\mathbf{C}(\boldsymbol{x},\boldsymbol{y})$ is monotone if the negations in $\mathbf{C}$ are only applied to variables from $\boldsymbol{y}$.

Lemma 5. *If a Boolean function f is computable by a (monotone) hypergraph program of size N then it can also be computed by a (monotone) nondeterministic circuit of size poly(N).*

Proof. Given a hypergraph program, we construct a nondeterministic circuit $\mathbf{C}(\boldsymbol{x},\boldsymbol{y})$. Its $\boldsymbol{x}$-variables are the variables of the program, and its advice variables correspond to the edges of the program. The circuit $\mathbf{C}$ will return 1 on $(\boldsymbol{x},\boldsymbol{y})$ iff the family $\{e_i \mid \boldsymbol{y}_i = 1\}$ of edges of the hypergraph forms an independent set covering all zeros. It is easy to construct a polynomial-size circuit checking this property. Indeed, for each pair of intersecting edges e_i, e_j, it is enough to compute $\neg e_i \lor \neg e_j$ and for each vertex of the hypergraph labelled by t and adjacent to edges $e_1, \dots, e_k$ to compute $t \lor \bigvee_{i=1}^k e_i$. It then remains to take a conjunction of all computed expressions.

It is easy to see that the resulting nondeterministic circuit is monotone if hypergraph program is monotone. ❑

Lemma 6. *If f is computable by a (monotone) nondeterministic Boolean circuit of size N then it can also be computed by a (monotone) hypergraph program of size poly(N) and degree ≤ 3.*

Proof. Let f be computed by a nondeterministic circuit $\mathbf{C}(\boldsymbol{x},\boldsymbol{y})$ with input variables $\boldsymbol{x}$ and advice variables $\boldsymbol{y}$.

Let $g_1, \dots, g_n$ be the nodes of $\mathbf{C}$ (including the $\boldsymbol{x}$ and $\boldsymbol{y}$). For each node g_i, we take a vertex g labelled by 0 in the hypergraph and a pair of hyperedges $\bar{e}_{g_i}$

and e_{g_i}, both containing g_i. No other edge will contain g_i and thus either $\bar{e}_{g_i}$ or e_{g_i} should be present in any cover of zeros. Intuitively, given an input, if a node g_i is positive then e_{g_i} will belong to the cover; otherwise, $\bar{e}_{g_i}$ will be there.

To ensure this property, for each input variable x_i, we add a fresh vertex labelled by $\neg x_i$ to e_{x_i} and a fresh vertex labelled by x_i to $\bar{e}_{x_i}$. For each gate g_i, we consider three cases.

– If $g_i = \neg g_j$ then we add a fresh vertex labelled by 1 to both e_{g_i} and $\bar{e}_{g_j}$ and a fresh vertex labelled by 1 to both $\bar{e}_{g_i}$ and e_{g_j}.
– If $g_i = g_j \vee g_k$ then we add a fresh vertex labelled by 1 to e_{g_j} and $\bar{e}_{g_i}$, add a fresh vertex labelled by 1 to e_{g_k} and $\bar{e}_{g_i}$. Then we add fresh vertices h_j and h_k labelled by 1 to $\bar{e}_{g_j}$ and $\bar{e}_{g_k}$, respectively, and a fresh vertex u_i labeled by 0 to $\bar{e}_{g_i}$. Finally, we add two new hyperedges $\{h_j, u_i\}$ and $\{h_k, u_i\}$.
– If $g_i = g_j \wedge g_k$ then we use the dual of the construction above.

It is not hard to see that e_{g_i} is in the cover iff it contains $\bar{e}_{g_j}$ and that e_{g_i} is in the cover iff it contains either e_{g_j} or e_{g_k}. Indeed, if, say, the cover contains e_{g_j} then it cannot contain $\bar{e}_{g_i}$. On the other hand, if $\bar{e}_{g_j}$ is not in the cover then we can add the hyperedge $\{h_j, u_i\}$ to cover u_i. Conversely, if neither e_{g_j} nor e_{g_k} is in the cover, then it must contain both $\bar{e}_{g_j}$ and $\bar{e}_{g_k}$ and so, neither $\{h_j, u_i\}$ nor $\{h_k, u_i\}$ can belong to the cover and we will have to include $\bar{e}_{g_i}$ to the cover.

Finally we add one more fresh vertex labelled by 0 to the edge e_g that corresponds to the output gate of $\mathbf{C}$. It is easy to see by induction on the structure of $\mathbf{C}$ that, for each $\boldsymbol{x}$ there is $\boldsymbol{y}$ such that $\mathbf{C}(\boldsymbol{x}, \boldsymbol{y}) = 1$ iff the hypergraph program returns 1 on $\boldsymbol{x}$.

If $\mathbf{C}$ is a monotone Boolean circuit, then we remove all vertices labeled by $\neg x_i$. It should be clear that there is a cover in the hypergraph on given $\boldsymbol{x}$ iff there are $\boldsymbol{y}$ and $\boldsymbol{x}' \leq \boldsymbol{x}$ such that $\mathbf{C}(\boldsymbol{x}', \boldsymbol{y}) = 1$. ❑

8 The Size of Rewritings over TBoxes of Depth 1

We are now in a position to apply the machinery developed above. Our aim in this section is to show that all CQs over TBoxes of depth 1 can be rewritten as polynomial-size NDL-queries, but not as polynomial-size PE-queries. On the other hand, *tree-shaped* CQs over TBoxes of depth 1 always have polynomial-size PE-rewritings.

Theorem 6. *For any CQ* $\boldsymbol{q}$ *and any TBox* $\mathcal{T}$ *of depth 1, there is an NDL-rewriting of* $\boldsymbol{q}$ *and* $\mathcal{T}$ *of polynomial size.*

Proof. Take a CQ $\boldsymbol{q}$ and a TBox $\mathcal{T}$ of depth 1. By Theorem 3, the hypergraph $H^{\boldsymbol{q}}_{\mathcal{T}}$ is of degree ≤ 2, and so, by Lemma 1 and 2, there is a hypergraph program of degree 2 computing $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ of size polynomial in $|\boldsymbol{q}|$. By Lemma 4, we have a monotone NBP of polynomial size computing $f^*_{H^{\boldsymbol{q}}_{\mathcal{T}}}$. But then we also have a polynomial-size monotone Boolean circuit that computes $f^*_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ (see, e.g., [18]). By replacing $\wedge$ with $\vee$, and $\vee$ with $\wedge$ in this circuit, we obtain a monotone circuit computing $f_{H^{\boldsymbol{q}}_{\mathcal{T}}}$ whose size is polynomial in $|\boldsymbol{q}|$. It remains to apply Theorem 2 and Proposition 1. ❑

On the other hand, our next theorem shows that there exist CQs and TBoxes of depth 1 for which there are no polynomial-size PE-rewritings:

Theorem 7. *There is a sequence of CQs $\boldsymbol{q}_n$ and TBoxes $\mathcal{T}_n$ of depth 1 such that any PE-rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ (over H-complete ABoxes) is of size $2^{\Omega(\log^2 n)}$.*

Proof. It is known [10] that there exists a sequence of Boolean functions $f_n(\boldsymbol{x})$ that are computable by polynomial-size monotone NBPs, but any monotone Boolean formulas computing f_n are of size $2^{\Omega(\log^2 n)}$. (Grigni and Sipser [10] consider $f_n(\boldsymbol{x})$ that takes the adjacency matrix of a directed graph of n vertices with a distinguished vertex s as input and returns 1 iff there is a directed path from s to some vertex of outdegree at least two.)

We apply Lemmas 3 and 2 to $f_n(\boldsymbol{x})$ and obtain a sequence H'_n of polynomial-size monotone hypergraph programs of degree 2 computing f^*_n. Then we apply Theorem 4 to the hypergraph H_n of each H'_n and obtain a sequence of CQs $\boldsymbol{q}_n$ and TBoxes $\mathcal{T}_n$ such that H_n is isomorphic to $H^{\boldsymbol{q}_n}_{\mathcal{T}_n}$. We show that any PE-rewriting $\boldsymbol{q}'_n$ of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ can be transformed into a monotone Boolean formula computing f_n and having size $\leq |\boldsymbol{q}'_n|$.

To define χ_n, we eliminate the quantifiers in $\boldsymbol{q}'_n$ in the following way: take a constant a and replace every subformula of the form $\exists x\, \psi(x)$ in $\boldsymbol{q}'_n$ with $\psi(a)$, repeating this operation as long as possible. The resulting formula $\boldsymbol{q}''_n$ is built from atoms of the form $A_e(a)$, $R_v(a,a)$ and $S_e(a,a)$ using $\land$ and $\lor$. For every ABox $\mathcal{A}$ with a single individual a, we have $(\mathcal{T}_n, \mathcal{A}) \models \boldsymbol{q}_n$ iff $\mathcal{A} \models \boldsymbol{q}''_n$. Let χ_n be the result of replacing $S_e(a,a)$ in $\boldsymbol{q}''_n$ with $\bot$, $A_e(a)$ with p_e and $R_v(a,a)$ with p_v. Clearly, $|\chi_n| \leq |\boldsymbol{q}'_n|$. By the definition of $\mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ and Theorem 5, we obtain:

$$\chi_n(\boldsymbol{\alpha},\boldsymbol{\beta}) = 1 \quad \text{iff} \quad \mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}} \models \boldsymbol{q}''_n \quad \text{iff} \quad (\mathcal{T}_n, \mathcal{A}_{\boldsymbol{\alpha},\boldsymbol{\beta}}) \models \boldsymbol{q}_n \quad \text{iff} \quad f_{H_n}(\boldsymbol{\alpha},\boldsymbol{\beta}) = 1.$$

As H'_n computes f^*_n, we can obtain f^*_n from f_{H_n} by replacing each p_e with 1 and each p_v with the label of v in H'_n . The same substitution in χ_n (with $\top$ and $\bot$ in place of 1 and 0) gives a monotone formula that computes f^*_n. By swapping $\lor$ and $\land$ in it, we obtain a monotone formula χ'_n computing f_n. It remains to recall that $|\boldsymbol{q}'_n| \geq |\chi'_n| \geq 2^{\Omega(\log^2 n)}$. ❑

It may be of interest to note that the function f_n in the proof above is in the complexity class NLogSpace. The algorithm computing this function by querying the NDL-rewriting of Theorem 6 over single-individual ABoxes runs in polynomial time; but if such an algorithm uses any PE-rewriting then, by Theorem 7, superpolynomial time is required.

Note also that, by Theorem 3, the number of distinct tree witnesses for $\boldsymbol{q}_n$ and $\mathcal{T}_n$ does not exceed the number of variables in $\boldsymbol{q}_n$. However, many of these tree-witnesses are inconsistent with each other, which results in long PE-rewrtings. Such a situation can never happen for *tree-shaped* CQs.

8.1 Tree-Shaped Queries and TBoxes of Depth 1

Let $\boldsymbol{q}$ be a CQ and $\boldsymbol{z}$ a subset of the set of existentially quantified variables in $\boldsymbol{q}$. By a *$\boldsymbol{z}$-partition* of $\boldsymbol{q}$ we understand any disjoint sets of atoms $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$, called

$\boldsymbol{z}$*-components*, that cover $\boldsymbol{q}$ and are such that if, for $i = 1, 2$, each of $S_i(\boldsymbol{z}_i) \in \boldsymbol{q}$ contains a variable $z_i \in \boldsymbol{z}$ and z_1 is connected to z_2 by a path coming through variables in $\boldsymbol{z}$ only, then both $S_i(\boldsymbol{z}_i)$ are in the same component. Note that, for any $\boldsymbol{z}$-partition of $\boldsymbol{q}$, every tree witness for $\exists \boldsymbol{z}\, \boldsymbol{q}$ and any $\mathcal{T}$ is contained in some $\boldsymbol{z}$-component of the partition.

Given a TBox $\mathcal{T}$ and a CQ $\boldsymbol{q}(\boldsymbol{x}) = \exists \boldsymbol{y}\, \varphi(\boldsymbol{x}, \boldsymbol{y})$, we define a CQ $\boldsymbol{q}^\dagger(\boldsymbol{x}) = \exists \boldsymbol{y}\, \boldsymbol{q}^{\boldsymbol{y}}$, where $\boldsymbol{q}^{\boldsymbol{z}}$ is computed recursively as follows: we take a $\boldsymbol{z}$-partition $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$ of $\boldsymbol{q}$, take $\boldsymbol{z}_j$ to be the set of the variables in $\boldsymbol{z}$ that occur in $\boldsymbol{q}_j$, for each $j = 1, \dots, k$, ($\boldsymbol{z}_1, \dots, \boldsymbol{z}_k$ form a partition of $\boldsymbol{z}$) and set $\boldsymbol{q}^{\boldsymbol{z}} = \boldsymbol{q}_1^{*\boldsymbol{z}_1} \land \dots \land \boldsymbol{q}_k^{*\boldsymbol{z}_k}$, where

$$\boldsymbol{q}_j^{*\boldsymbol{z}_j} = \begin{cases} (\boldsymbol{q}_j)^{\boldsymbol{z}_j \setminus \{z_j\}} \quad \lor \displaystyle\bigvee_{\substack{\text{tree witness } \mathfrak{t} \text{ for } \exists \boldsymbol{z}_j\, \boldsymbol{q}_j \text{ and } \mathcal{T} \\ \text{with } z_j \in \mathfrak{t}_\mathsf{i}}} \big(\mathsf{tw}_\mathfrak{t} \land (\boldsymbol{q}_j \setminus \boldsymbol{q}_\mathfrak{t})^{\boldsymbol{z}_j \setminus (\mathfrak{t}_\mathsf{r} \cup \mathfrak{t}_\mathsf{i})}\big), & \text{if there is } z_j \in \boldsymbol{z}_j, \\ \boldsymbol{q}_j, & \text{otherwise.} \end{cases}$$

Note that $\boldsymbol{q}^{\boldsymbol{z}}$ depends on the choice of $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$ and $z_j \in \boldsymbol{z}_j$, which can be arbitrary. Intuitively, the first disjunct of $\boldsymbol{q}^{\boldsymbol{z}}$ reflects the situation where z_j is mapped to an ABox element; so we treat z_j as a free variable when rewriting $\boldsymbol{q}_j$. The other disjuncts reflect the case when z_j is mapped to the non-ABox part of the canonical model, in which case z_j belongs to the internal part $\mathfrak{t}_\mathsf{i}$ of a tree witness $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ for $\exists \boldsymbol{z}_j\, \boldsymbol{q}_j$ and $\mathcal{T}$. As the variables in $\mathfrak{t}_\mathsf{r}$ must be mapped to ABox elements, this leaves the set $\boldsymbol{q}_j \setminus \boldsymbol{q}_\mathfrak{t}$ of atoms with existentially quantified $\boldsymbol{z}_j \setminus (\mathfrak{t}_\mathsf{r} \cup \mathfrak{t}_\mathsf{i})$ for further rewriting (this set of variables does not contain z_j).

Theorem 8. *For any ABox $\mathcal{A}$ that is H-complete with respect to $\mathcal{T}$ and any $\boldsymbol{a} \subseteq \mathsf{ind}(\mathcal{A})$, we have $\mathcal{C}_{\mathcal{T},\mathcal{A}} \models \boldsymbol{q}(\boldsymbol{a})$ iff $\mathcal{A} \models \boldsymbol{q}^\dagger(\boldsymbol{a})$.*

Example 4. Take $\boldsymbol{q}$ and $\mathcal{T}$ from Example 1. The only $\{y_2, y_3\}$-component of $\boldsymbol{q}$ is $\boldsymbol{q}$. Then we pick, say y_2, and obtain $\boldsymbol{q}^{\{y_2, y_3\}} = \boldsymbol{q}^{\{y_3\}} \lor \big(\mathsf{tw}_{\mathfrak{t}^1} \land R(y_3, x_4)^\emptyset\big)$. Now, $\boldsymbol{q}$ has two $\{y_3\}$-components, $\{R(x_1, y_2)\}$ and $\boldsymbol{q}_1 = \{R(y_3, y_2), R(y_3, x_4)\}$. The former gives $R(x_1, y_2)$, while in the latter we have to pick y_3 and obtain $\boldsymbol{q}_1^\emptyset \lor \mathsf{tw}_{\mathfrak{t}^2}$, assuming that the empty set of atoms is $\top$. This gives the rewriting:

$$\boldsymbol{q}^\dagger(x_1, x_4) = \exists y_2, y_3 \Big[\Big(R(x_1, y_2) \land \big(\big(R(y_3, y_2) \land R(y_3, x_4)\big) \lor \mathsf{tw}_{\mathfrak{t}^2}\big)\Big) \lor \big(\mathsf{tw}_{\mathfrak{t}^1} \land R(y_3, x_4)\big)\Big].$$

A CQ $\boldsymbol{q}$ is said to be *tree-shaped* if its primal graph is a tree. In each component $\boldsymbol{q}_j$ of a tree-shaped CQs $\boldsymbol{q}$, we can choose a variable z_j that splits it in half. More formally, we have the following:

Proposition 2. *For any tree $T = (V, E)$, there is a vertex $v \in V$ such that each connected component obtained by removing v from T contains $\leq \frac{|V|}{2}$ vertices.*

By Proposition 2, any tree-shaped CQ can be split into components each of which contains less than a half of atoms of the CQ. By applying this argument recursively and using the fact that, if a TBox $\mathcal{T}$ is of depth 1 then, for any variable z in $\boldsymbol{q}$, the number of tree witnesses $\mathfrak{t} = (\mathfrak{t}_\mathsf{r}, \mathfrak{t}_\mathsf{i})$ for $\boldsymbol{q}$ and $\mathcal{T}$ with $z \in \mathfrak{t}_\mathsf{i}$ does not exceed 2, we obtain our final result:

Theorem 9. *Any tree-shaped CQ over any TBox of depth one has a polynomial PE-rewriting.*

Proof. We show that the rewriting $\boldsymbol{q}^\dagger$ is of size $O(|\boldsymbol{q}|^2 \cdot |\mathcal{T}|)$. We will assume (without loss of generality) that each formula $\mathsf{tw}_\mathfrak{t}$ is of the form

$$\mathsf{tw}_\mathfrak{t} \quad = \bigwedge_{x \in \mathfrak{t}_\mathsf{r} \setminus \{x_0\}} (x = x_0) \quad \wedge \quad \bigvee_{R \in \Omega_\mathfrak{t}} A_R(x_0),$$

where x_0 is a distinguished term in $\mathfrak{t}_\mathsf{r}$. Thus, the length of each tw_t does not exceed $|\mathcal{T}| \cdot (|\mathfrak{t}_\mathsf{r}| - 1)$. Denote by $F(n)$ the maximal size of $|\boldsymbol{q}^{\boldsymbol{z}}|$ for sets $\boldsymbol{z}$ and CQs $\boldsymbol{q}$ with at most n atoms. We claim that $F(n) \leq |\mathcal{T}| \cdot n^2$.

The proof is by induction and the indiction basis is clear. For the inductive step, observe that since the TBox $\mathcal{T}$ is of depth 1, each variable z can belong to $\mathfrak{t}_\mathsf{i}$ in at most one tree witness $\mathfrak{t}$ for $\boldsymbol{q}$ and $\mathcal{T}$. Thus, by the definition of $\boldsymbol{q}^{\boldsymbol{z}}$, for each $\boldsymbol{z}$-component with n_j atoms we have a formula of the length that does not exceed $F(n_j) + F(n_j - m_j) + |\mathcal{T}| \cdot (m_j - 1)$, where m_j, $1 \leq m_j \leq n_j$, is the number of terms in $\mathfrak{t}_\mathsf{r}$ for the tree witness (note that each term of $\mathfrak{t}_\mathsf{r}$ occurs in at least one atom in $\boldsymbol{q}_\mathfrak{t}$). By the induction hypothesis, we have

$$F(n_j - m_j) + |\mathcal{T}| \cdot (m_j - 1) \ \leq\ |\mathcal{T}| \cdot (n_j - m_j)^2 + |\mathcal{T}| \cdot (m_j - 1) \ \leq\ |\mathcal{T}| \cdot n_j^2.$$

So, by Proposition 2, we can partition $\boldsymbol{q}$ with n atoms into $\boldsymbol{z}$-components $\boldsymbol{q}_1, \dots, \boldsymbol{q}_k$ such that $\sum_{j=1}^k n_j = n$ and each $n_j \leq n/2$, where n_j is the number of atoms in $\boldsymbol{q}_j$. Then we have

$$F(n) \leq \sum_{j=1}^k \big(F(n_j) + |\mathcal{T}| \cdot n_j^2\big) \ \leq\ \sum_{j=1}^k 2|\mathcal{T}| \cdot n_j^2 \ \leq\ 2|\mathcal{T}| \cdot n/2 \cdot \sum_{j=1}^k n_j = |\mathcal{T}| \cdot n^2.$$

This finishes the proof of the theorem. ❑

9 Lower Bounds for Rewritings over TBoxes of Depth 2

As we saw above, CQs and TBoxes of depth 1 can be used to compute monotone Boolean functions, and the computational power of this (exotic) formalism is the same as that of monotone hypergraph programs of degree 2. In this section, we show that CQs and TBoxes of depth 2, as well as monotone hypergraphs of degree 3, can compute more complex Boolean functions, in particular, the NP-complete function checking whether a graph with n vertices contains a k-clique.

We remind the reader (see, e.g., [2] for details) that the monotone Boolean function $\textsc{Clique}_{n,k}(\boldsymbol{e})$ of $n(n-1)/2$ variables $e_{jj'}$, $1 \leq j < j' \leq n$, returns 1 iff the graph with vertices $\{1, \dots, n\}$ and edges $\{\{i, j\} \mid e_{jj'} = 1\}$ contains a k-clique. Clearly, $\textsc{Clique}_{n,k}$ is NP-complete. A series of papers, started by Razborov's [17], gave an exponential lower bound for the size of monotone circuits computing $\textsc{Clique}_{n,k}$: $2^{\Omega(\sqrt{k})}$ for $k \leq \frac{1}{4}(n/\log n)^{2/3}$ [1]. For monotone formulas, an even better lower bound is known: $2^{\Omega(k)}$ for $k = 2n/3$ [16].

In this section, we first construct a monotone hypergraph program that computes the function $\textsc{Clique}_{n,k}$ and then use the intuition behind the construction to encode $\textsc{Clique}_{n,k}$ by means of a Boolean CQ $\boldsymbol{q}_{n,k}$ and a TBox $\mathcal{T}_{n,k}$ of polynomial size. As a consequence, any PE- or NDL-rewriting of $\boldsymbol{q}_{n,k}$ and $\mathcal{T}_{n,k}$ is of exponential size, while any FO-rewriting is of superpolynomial size unless $\mathrm{NP} \subseteq \mathrm{P/poly}$.

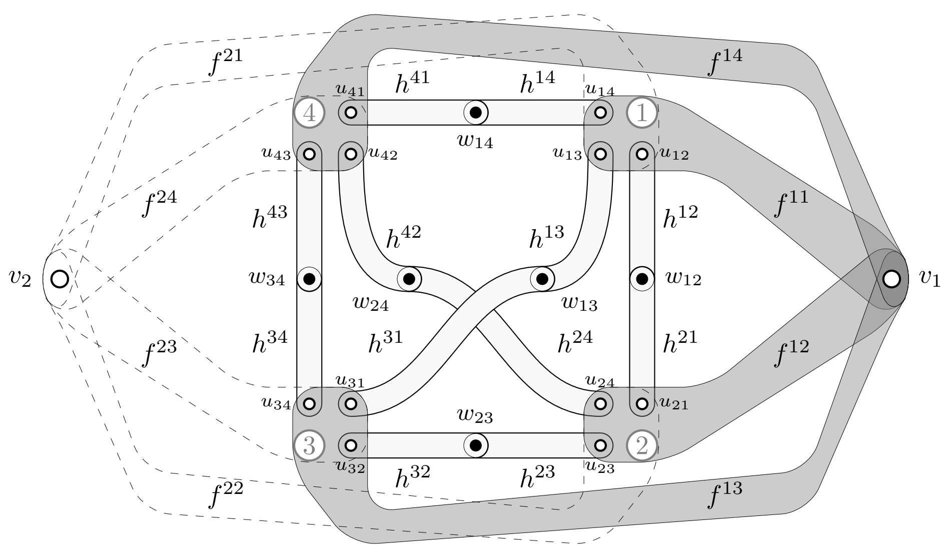
Given n and k as above, let $H_{n,k}$ be a monotone hypergraph program with the vertices

- v_i labelled by 0, for $1 \leq i \leq k$,
- $w_{jj'}$ labelled by the propositional variable $e_{jj'}$, for $1 \leq j < j' \leq n$,
- $u_{jj'}$ labelled by 1, for $1 \leq j \neq j' \leq n$,

and the hyperedges

- $f^{ij} = \{v_i\} \cup \{u_{jj'} \mid 1 \leq j' \leq n, j' \neq j\}$, for $1 \leq i \leq k$ and $1 \leq j \leq n$,
- $h^{jj'} = \{w_{jj'}, u_{jj'}\}$ and $h^{j'j} = \{w_{jj'}, u_{j'j}\}$, for $1 \leq j < j' \leq n$.

Informally, the vertices v_i of the hypergraph $H_{n,k}$ represent a k-clique in a given graph with n vertices. The vertices $w_{jj'}$ represent the edges of the complete graph with n vertices; they can be turned 'on' or 'off' by means of the Boolean variables $e_{jj'}$. The vertex $u_{jj'}$ together with the hyperedge $h^{jj'}$ represent the 'half' of the edge connecting j and j' that is adjacent to j. The edges f^{ij} correspond to the choice of the jth vertex of the graph as the ith vertex in the clique. The hypergraph $H_{4,2}$ is shown below:



Theorem 10. *The hypergraph programs $H_{n,k}$ compute $\textsc{Clique}_{n,k}$.*

Proof. We have to show that, for each Boolean vector $\boldsymbol{e} \in \{0,1\}^{n(n-1)/2}$, there is an independent set X of hyperedges covering all 0s in $H_{n,k}$ iff $\textsc{Clique}_{n,k}(\boldsymbol{e}) = 1$.

(⇐) Let $\lambda\colon \{1,\dots,k\} \to \{1,\dots,n\}$ be such that $C = \{\lambda(i) \mid 1 \le i \le k\}$ is a k-clique in the graph, G, given by $\boldsymbol{e}$. We claim that

$$X = \{f^{i\lambda(i)} \mid 1 \le i \le k\} \cup \{h^{jj'} \mid j \notin C, j' \in C\} \cup \{h^{jj'} \mid j, j' \notin C \text{ and } j < j'\}$$

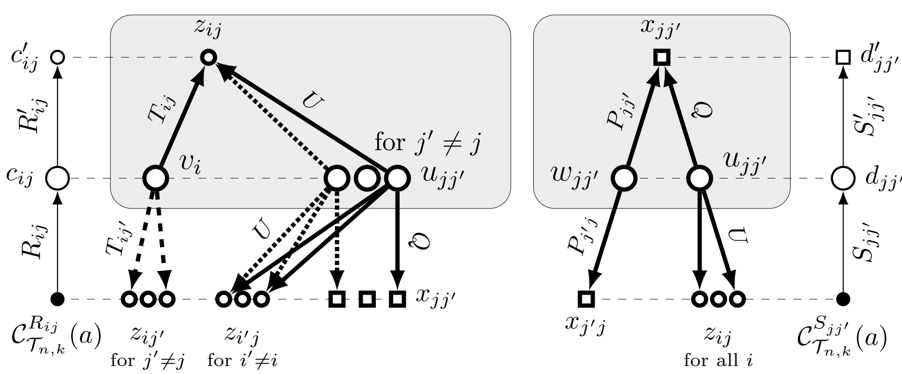
is independent and covers all vertices labelled by zeros. Indeed, X is independent because, in every $h^{jj'} \in X$, the index j does not belong to C. By definition, each $f^{i\lambda(i)}$ covers v_i, for $1 \le i \le k$. Thus, it remains to show that any $w_{jj'}$ with $e_{jj'} = 0$ (that is, the edge $\{j, j'\}$ belongs to the complement of G) is covered by some hyperedge. All edges of the complement of G can be divided into two groups: those that are adjacent to C, and those that are not. The $w_{jj'}$ that correspond to the edges of the former group are covered by the $h^{jj'}$ from the middle disjunct of X, where j corresponds to the end of the edge $\{j, j'\}$ that is not C. To cover $w_{jj'}$ of the latter group, take $h^{jj'}$ from the last disjunct of X.

(⇒) Suppose X is an independent set covering all zeros labelling the vertices of the hypergraph $H_{n,k}$, for an input $\boldsymbol{e}$. The vertex v_i, $1 \le i \le k$, is labelled by 0, and so there is $\lambda(i)$ such that $f^{i\lambda(i)} \in X$. We claim that $C = \{\lambda(i) \mid 1 \le i \le k\}$ is a k-clique. Indeed, suppose that the graph given by $\boldsymbol{e}$ contains no edge between some vertices $j, j' \in C$, that is, $e_{jj'} = 0$ for $j < j'$. Since $w_{jj'}$ is labelled by 0, it must be covered by a hyperedge in X, which can only be either $h^{jj'}$ or $h^{j'j}$ (see the picture above). But $h^{jj'}$ intersects $f^{\lambda^{-1}(j)j}$ and $h^{j'j}$ intersects $f^{\lambda^{-1}(j')j'}$, which is a contradiction. ❑

We are now in a position to define the Boolean CQ $\boldsymbol{q}_{n,k}$ and the TBox $\mathcal{T}_{n,k}$ of polynomial size (in n) that can compute $\textsc{Clique}_{n,k}$. Let $\boldsymbol{q}_{n,k}$ contain the following atoms:

$T_{ij}(v_i, z_{ij})$,	for $1 \le i \le k$ and $1 \le j \le n$,
$P_{jj'}(w_{jj'}, x_{jj'})$, $P_{j'j}(w_{jj'}, x_{j'j})$,	for $1 \le j < j' \le n$,
$Q(u_{jj'}, x_{jj'})$, $U(u_{jj'}, z_{ij})$,	for $1 \le j \ne j' \le n$ and $1 \le i \le k$.

The picture below illustrates the fragments of $\boldsymbol{q}_{n,k}$ centred around each variable of the form z_{ij} and $x_{jj'}$ (the fragment centred around $x_{j'j}$ is similar to that of $x_{jj'}$ except the index of the $w_{jj'}$):



The whole CQ $\boldsymbol{q}_{n,k}$ is illustrated by the picture in Appendix A.

The TBox $\mathcal{T}_{n,k}$ mimics the arrangement of atoms in the layers depicted above and contains the following inclusions: for $1 \leq i \leq k$ and $1 \leq j \neq j' \leq n$,

$$\begin{array}{lllll}
A_{ij} \equiv \exists R_{ij}, & R_{ij} \sqsubseteq T^-_{ij'}, & R_{ij} \sqsubseteq U^-, & R_{ij} \sqsubseteq Q^-, & \exists R^-_{ij} \sqsubseteq A'_{ij} \\
A'_{ij} \equiv \exists R'_{ij}, & R'_{ij} \sqsubseteq T_{ij}, & R'_{ij} \sqsubseteq U, & & \\
B_{jj'} \equiv \exists S_{jj'}, & S_{jj'} \sqsubseteq P^-_{j'j}, & S_{jj'} \sqsubseteq U^-, & & \exists S^-_{jj'} \sqsubseteq B'_{jj'}, \\
B'_{jj'} \equiv \exists S'_{jj'}, & S'_{jj'} \sqsubseteq P_{jj'}, & S'_{jj'} \sqsubseteq Q. & &
\end{array}$$

The picture above also shows the elements and 'generating roles' of the models $\mathcal{C}^{R_{ij}}_{\mathcal{T}_{n,k}}(a)$ and $\mathcal{C}^{S_{jj'}}_{\mathcal{T}_{n,k}}(a)$. The omitted roles are uniquely determined by the role inclusions in $\mathcal{T}_{n,k}$. Those roles, in fact, appear in the respective layer of the depicted CQ fragments, while the horizontal dashed lines show possible ways of embedding the fragments of $\boldsymbol{q}_{n,k}$ into the respective canonical models. These embeddings give rise to the following tree witnesses for $\boldsymbol{q}_{n,k}$ and $\mathcal{T}_{n,k}$:

- $\mathfrak{t}^{ij} = (\mathfrak{t}^{ij}_\mathsf{r}, \mathfrak{t}^{ij}_\mathsf{i})$ generated by R_{ij}, for $1 \leq i \leq k$ and $1 \leq j \leq n$, where

$$\begin{array}{lcl}
\mathfrak{t}^{ij}_\mathsf{r} & = & \{z_{ij'}, x_{jj'} \mid 1 \leq j' \leq n,\ j' \neq j\} \ \cup\ \{z_{i'j} \mid 1 \leq i' \leq k,\ i \neq i'\}, \\
\mathfrak{t}^{ij}_\mathsf{i} & = & \{v_i, z_{ij}\} \cup \{u_{jj'} \mid 1 \leq j' \leq n,\ j' \neq j\};
\end{array}$$

- $\mathfrak{s}^{jj'} = (\mathfrak{s}^{jj'}_\mathsf{r}, \mathfrak{s}^{jj'}_\mathsf{i})$ and $\mathfrak{s}^{j'j} = (\mathfrak{s}^{j'j}_\mathsf{r}, \mathfrak{s}^{j'j}_\mathsf{i})$, generated by $S_{jj'}$ and $S_{j'j}$, where

$$\begin{array}{ll}
\mathfrak{s}^{jj'}_\mathsf{r} = \{x_{j'j}\} \cup \{z_{ij} \mid 1 \leq i \leq k\}, & \mathfrak{s}^{j'j}_\mathsf{r} = \{x_{jj'}\} \cup \{z_{ij'} \mid 1 \leq i \leq k\}, \\
\mathfrak{s}^{jj'}_\mathsf{i} = \{w_{jj'}, u_{jj'}, x_{jj'}\}, & \mathfrak{s}^{j'j}_\mathsf{i} = \{w_{jj'}, u_{j'j}, x_{j'j}\}.
\end{array}$$

The tree witnesses $\mathfrak{t}^{ij}$, $\mathfrak{s}^{jj'}$ and $\mathfrak{s}^{j'j}$ are uniquely determined by their most remote (from the root) variable, z_{ij}, $x_{jj'}$ and $x_{j'j}$, respectively, and correspond to the hyperedges f^{ij}, $h^{jj'}$, $h^{j'j}$ of the hypergraph $H_{n,k}$; their internal variables of the form v_i, $w_{jj'}$ and $u_{jj'}$ correspond to the vertices in the respective hyperedge.

Given a Boolean vector $\boldsymbol{e}$ representing a graph with n vertices, we construct an ABox $\mathcal{A}_{\boldsymbol{e}}$ with a single individual a and the following atoms:

$$\begin{array}{l}
A(a), \quad Q(a,a), \quad U(a,a), \\
\qquad\qquad\qquad P_{jj'}(a,a) \text{ and } P_{j'j}(a,a), \text{ for } 1 \leq j < j' \leq n \text{ with } e_{jj'} = 1.
\end{array}$$

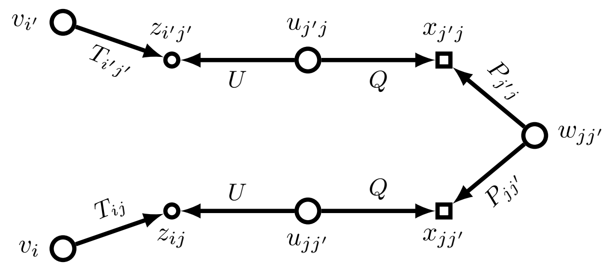
Lemma 7. $(\mathcal{T}_{n,k}, \mathcal{A}_{\boldsymbol{e}}) \models \boldsymbol{q}_{n,k}$ *iff* $\textsc{Clique}_{n,k}(\boldsymbol{e}) = 1$.

Proof. ($\Rightarrow$) Suppose $(\mathcal{T}_{n,k}, \mathcal{A}_{\boldsymbol{e}}) \models \boldsymbol{q}_{n,k}$. Then there is a homomorphism g from $\boldsymbol{q}_{n,k}$ to the canonical model $\mathcal{C}$ of $(\mathcal{T}_{n,k}, \mathcal{A}_{\boldsymbol{e}})$. Since the only points of $\mathcal{C}$ that belong to $\exists T_{ij}$ are of the form c_{ij} (in the picture above) and $\boldsymbol{q}_{n,k}$ contains atoms of the form $T_{ij}(v_i, z_{ij})$, there is a function $\lambda\colon \{1, \dots, k\} \to \{1, \dots, n\}$ such that $g(v_i) = c_{i\lambda(i)}$. We claim that $C = \{\lambda(i) \mid 1 \leq i \leq k\}$ is a k-clique in the graph given by $\boldsymbol{e}$.

We first show that $\lambda(i) \neq \lambda(i')$, for $1 \leq i \neq i' \leq k$. Indeed, otherwise we would have $\lambda(i) = \lambda(i') = j$, for some distinct i, i'. Since both $T_{ij}(v_i, z_{ij})$ and

$T_{i'j}(v_{i'}, z_{i'j})$ are in $\boldsymbol{q}_{n,k}$, we have $g(z_{ij}) = c'_{ij}$ and $g(z_{i'j}) = c'_{i'j}$. Take some $j' \neq j$. Since $U(u_{jj'}, z_{ij}), U(u_{jj'}, z_{i'j}) \in q_{n,k}$, we obtain $g(u_{jj'}) = c_{ij}$ and $g(u_{jj'}) = c_{i'j}$, contrary to $i \neq i'$.

Next, we show that $e_{jj'} = 1$, for all $j, j' \in C$ with $j < j'$. Since $U(u_{jj'}, z_{ij})$ is in $\boldsymbol{q}_{n,k}$, we have $g(u_{jj'}) = c_{ij}$, and so $g(x_{jj'}) = a$. Similarly, we also have $g(u_{j'j}) = c_{i'j'}$ and $g(x_{j'j}) = a$. Then, since $\boldsymbol{q}_{n,k}$ contains both $P_{jj'}(w_{jj'}, x_{jj'})$ and $P_{j'j}(w_{jj'}, x_{j'j})$ and $\mathcal{C}$ contains no pair of points in both $P_{jj'}$ and $P_{j'j}$ apart from (a, a), we obtain $e_{jj'} = 1$ whenever $g(x_{jj'}) = g(x_{j'j}) = a$ (see the picture below).



($\Leftarrow$) Suppose that $\lambda\colon \{1, \dots, k\} \to \{1, \dots, n\}$ is a k-clique and denote by C the set $\{\lambda(i) \mid 1 \leq i \leq k\}$. We construct a homomorphism g from $\boldsymbol{q}_{n,k}$ to the canonical model of $(\mathcal{T}_{n,k}, \mathcal{A}_{\boldsymbol{e}})$ by taking, for $1 \leq i \leq k$ and $1 \leq j < j' \leq n$,

$$g(v_i) = c_{i\lambda(i)},$$

$$g(z_{ij}) = \begin{cases} c'_{ij}, & \text{if } j = \lambda(i), \\ a & \text{otherwise,} \end{cases} \qquad g(w_{jj'}) = \begin{cases} a, & \text{if } j, j' \in C, \\ d_{j'j}, & \text{if } j' \notin C \text{ and } j \in C, \\ d_{jj'}, & \text{otherwise,} \end{cases}$$

and, for $1 \leq j \neq j' \leq n$,

$$g(u_{jj'}) = \begin{cases} c_{\lambda^{-1}(j)j}, & \text{if } j \in C, \\ d_{jj'}, & \text{if } j \notin C, j' \in C, \\ d_{jj'}, & \text{if } j, j' \notin C,\ j < j', \\ a, & \text{if } j, j' \notin C,\ j' < j, \end{cases} \qquad g(x_{jj'}) = \begin{cases} a, & \text{if } j \in C, \\ d'_{jj'}, & \text{if } j \notin C, j' \in C, \\ d'_{jj'}, & \text{if } j, j' \notin C,\ j < j', \\ a, & \text{if } j, j' \notin C,\ j' < j. \end{cases}$$

This homomorphism mimics the cover X constructed for $H_{n,k}$ in the proof of Theorem 10. For example, in the definition of $g(u_{jj'})$, the first case corresponds to $u_{jj'} \in f^{\lambda^{-1}(j)j} \in X$; the second and third cases to $u_{jj'} \in h^{jj'} \in X$; and in the fourth case, $u_{jj'}$ is not covered by X. It follows that $(\mathcal{T}_{n,k}, \mathcal{A}_{\boldsymbol{e}}) \models \boldsymbol{q}_{n,k}$. ❑

Theorem 11. *There exists a sequence of CQs $\boldsymbol{q}_n$ and TBoxes $\mathcal{T}_n$ of depth 2 such that any PE- and NDL-rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of exponential size, while any FO-rewriting of $\boldsymbol{q}_n$ and $\mathcal{T}_n$ is of superpolynomial size* (*unless* NP $\subseteq$ P/*poly*).

Proof. Given a PE-, FO- or NDL-rewriting $\boldsymbol{q}'_{n,k}$ of $\boldsymbol{q}_{n,k}$ and $\mathcal{T}_{n,k}$, we show how to construct, respectively, a monotone Boolean formula, a Boolean formula or a monotone Boolean circuit for the function CLIQUE$_{n,k}$ of size $|\boldsymbol{q}'_{n,k}|$.

Suppose first that $\boldsymbol{q}'_{n,k}$ is a PE-rewriting of $\boldsymbol{q}_{n,k}$ and $\mathcal{T}_{n,k}$. To begin with, we eliminate the quantifiers in $\boldsymbol{q}'_{n,k}$. Namely, we replace every subformula of the form $\exists x\, \psi(x)$ in $\boldsymbol{q}'_n$ with $\psi(a)$. In the resulting formula $\boldsymbol{q}''_{n,k}$, we replace each $P_{jj'}(a,a)$ and $P_{j'j}(a,a)$ by $e_{jj'}$, each $T_{ij}(a,a)$ by 0, each $U(a,a)$ and $Q(a,a)$ by 1, each $A_{ij}(a)$ and $B_{jj'}(a)$ by 1 and each $A'_{ij}(a)$ and $B'_{jj'}(a)$ by 0. One can check that the resulting propositional monotone Boolean formula computes $\textsc{Clique}_{n,k}$.

If $\boldsymbol{q}'_{n,k}$ is an FO-rewriting of $\boldsymbol{q}_{n,k}$, then we eliminate the quantifiers by replacing both $\exists x\, \psi(x)$ and $\forall x\, \psi(x)$ in $\boldsymbol{q}'_{n,k}$ with $\psi(a)$, and then carry out the replacing procedure described above, obtaining a propositional Boolean formula that computes $\textsc{Clique}_{n,k}$.

If $(\mathit{\Pi}, \boldsymbol{q}'_{n,k})$ is an NDL-rewriting of $\boldsymbol{q}_{n,k}$, we replace all the individual variables in $\mathit{\Pi}$ with a and then perform the replacement described above. Denote the resulting propositional NDL-program by $\mathit{\Pi}'$. The program $\mathit{\Pi}'$ can now be transformed into a monotone Boolean circuit computing $\textsc{Clique}_{n,k}$: for every (propositional) variable p occurring in the head of a clause in $\mathit{\Pi}'$, we introduce an $\vee$-gate whose output is p and inputs are the bodies of the clauses with the head p; and for each such body, we introduce an $\wedge$-gate whose inputs are the propositional variables in the body.

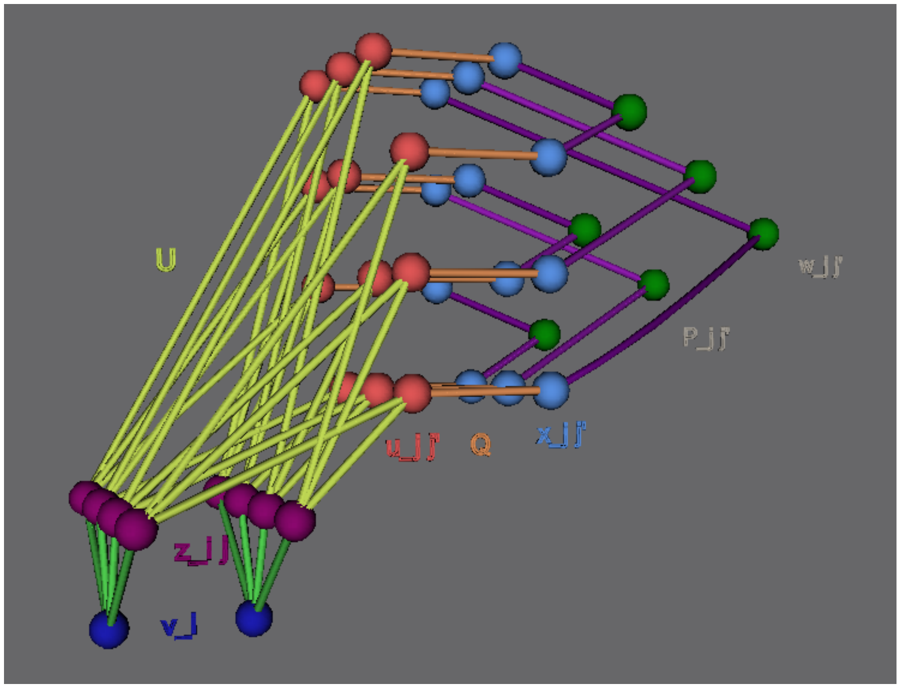
Now Theorem 11 follows from the lower bounds for monotone Boolean circuits and formulas computing $\textsc{Clique}_{n,k}$ given at the beginning of this section. ❑

10 Open Questions

Although the hypergraph technique developed in this paper proves to be fruitful and elegant, some natural questions in this area still remain open:

- Is it possible to obtain non-trivial upper and lower bounds for the size of PE-, FO- and NDL-rewritings for tree-shaped CQs and theories of bounded depth? (Note that in [12] we present exponential lower bounds for PE- and NDL- rewritings for tree-shaped queries and theories of unbounded depth.)
- Is it possible to obtain general 'representation theorems' for hypergraphs, similar to Theorem 5, which would witness the correspondence between classes of hypergraphs and classes of TBoxes?

A CQ $q_{n,k}$



References

1. N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.
2. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
3. 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.
4. B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.
5. J. Avigad. Eliminating definitions and Skolem functions in first-order logic. In *Proc. of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS'91)*, pages 139–146. IEEE Computer Society, 2001.
6. 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.
7. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.
8. 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. *J. of Automated Reasoning*, 39(3):385–429, 2007.
9. G. Gottlob and T. Schwentick. Rewriting ontological queries into small nonrecursive datalog programs. In *Proc. of the 24th Int. Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
10. M. Grigni and M. Sipser. Monotone separation of logarithmic space from logarithmic depth. *J. Comput. Syst. Sci.*, 50(3):433–437, 1995.
11. S. Jukna. *Boolean Function Complexity — Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.
12. S. Kikot, R. Kontchakov, 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)*, volume 7392 of *LNCS*, pages 263–274. Springer, 2012.
13. S. Kikot, R. Kontchakov, V. Podolskii, and M. Zakharyaschev. Long rewritings, short rewritings. In *Proc. of the 2012 Int. Workshop on Description Logics (DL 2012)*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
14. S. Kikot, R. Kontchakov, and M. Zakharyaschev. On (In)Tractability of OBDA with OWL 2 QL. In *Proc. of the 24th Int. Workshop on Description Logics (DL 2011)*, volume 745 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.
15. S. Kikot, R. Kontchakov, and M. Zakharyaschev. Conjunctive query answering with OWL 2 QL. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 13th Int. Conf. KR 2012*. AAAI Press, 2012.
16. R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.
17. A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.
18. 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 *LNCS*, pages 47–60. Springer, 1991.