Query Rewriting under $\mathcal{EL}$-TBoxes: Efficient Algorithms

Peter Hansen[1], Carsten Lutz[1], İnanç Seylan[1], and Frank Wolter[2]

[1] University of Bremen, Germany, {hansen,clu,seylan}@informatik.uni-bremen.de
[2] University of Liverpool, UK, frank@csc.liv.ac.uk

Abstract. We propose a new type of algorithm for computing first-order (FO) rewritings of concept queries under $\mathcal{EL}$-TBoxes that is tailored towards efficient implementation. The algorithm outputs a non-recursive datalog rewriting if the input is FO-rewritable and otherwise reports non-FO-rewritability. We carry out experiments with ontologies from practical applications which show that our algorithm performs very well in practice, and that $\mathcal{EL}$-TBoxes originating from real-world applications admit FO-rewritings (of reasonable size) in almost all cases, even when in theory such rewritings are not guaranteed to exist.

1 Introduction

Query rewriting is an important technique for implementing ontology-based data access (OBDA) based on a relational database system (RDBMS), thus taking advantage of those systems' efficiency and maturity [16, 10]. The general idea is to transform the original query q and the relevant TBox $\mathcal{T}$ into a first-order (FO) query $q_{\mathcal{T}}$ that is then handed over to the RDBMS for execution. One limitation of this approach is that, for the majority of description logics that are used as ontology languages, the query $q_{\mathcal{T}}$ is not guaranteed to exist. In fact, this is the case already for the members of the popular $\mathcal{EL}$ family of lightweight DLs [3, 12], which underly the OWL2 EL profile and are frequently used as ontology languages in health care and biology. This observation, however, does not rule out the possibility that FO-rewritings still exist in many practically relevant cases. In fact, TBoxes that emerge from practical applications tend to have a rather simple structure and one might thus speculate that, indeed, FO-rewritings under $\mathcal{EL}$-TBoxes tend to exist in practice.

In this paper, we consider the computation of FO-rewritings of concept queries under TBoxes that are formulated in the description logic $\mathcal{EL}$, where a concept query takes the form $C(x)$ with C an $\mathcal{EL}$-concept. It has recently been shown in [7] that deciding whether a concept query $C(x)$ is FO-rewritable under an $\mathcal{EL}$-TBox $\mathcal{T}$ is a PSpace-complete, and that the problem becomes ExpTime-complete when a signature is imposed on the set of admitted database instances (represented as an ABox). This shows that computing the desired rewritings is not an easy task. The existing approaches to query rewriting in $\mathcal{EL}$ and their relevance to the computation of FO-rewritings can be summarized as follows:

(i) approaches that target rewritings in the more expressive query language datalog which are incomplete in the sense that the generated datalog-rewritings are not guaranteed to be non-recursive even if there is an FO-rewriting [17, 15, 14]; (ii) backwards chaining approaches for existential rules (a strict generalization of $\mathcal{EL}$) which are complete in the sense that they find an FO-rewriting if there is one, but need not terminate otherwise [9]; (iii) the complete and terminating approach from [7] which aims at proving upper complexity bounds for FO-rewritings which cannot be expected to be feasible in practice because it relies on brute-force enumeration techniques.[3]

The aim of this paper is *to design algorithms for computing FO-rewritings of concept queries under $\mathcal{EL}$-TBoxes that are complete, terminating, and feasible in practice.* To this end, we start with a marriage of approaches (ii) and (iii) to get the best of both worlds; in particular, (ii) appears to be practically more feasible than (iii) while (iii) provides a way to achieve termination. The resulting algorithm is conceptually simple and constitutes a significant step towards our goal. However, it produces FO-rewritings that are unions of conjunctive queries (UCQs), which results in two significant drawbacks: first, recent experiments [11] have shown that executing UCQ-rewritings on RDBMSs is prohibitively expensive while executing equivalent rewritings that take the form of non-recursive datalog programs is much more feasible (even when the original query is a conjunctive query); and second, UCQ-rewritings can be of excessive size even in practically relevant cases [18].

To address these shortcomings, we refine our original algorithm to what we call a *decomposed algorithm*. While the original algorithm uses tree-shaped conjunctive queries (CQs) as an internal data structure, the new algorithm only represents single nodes of CQs together with information of how to reassemble these nodes into full tree-shaped CQs. This can be seen as a way to implement structure sharing and it also allows us to directly produce rewritings that are non-recursive datalog programs, avoiding UCQ-rewritings altogether. The algorithm runs in exponential time, is capable of deciding the existence of FO-rewritings in ExpTime, and produces monadic non-recursive datalog rewritings that are of at most exponential size (but much smaller in practice). Technically, the decomposed algorithm is much more subtle than the original one.

We then evaluate the decomposed algorithm by carrying out experiments with seven ontologies from practical applications. We ask for an FO-rewriting for every concept query $A(x)$, with A a concept name from the ontology under consideration. Out of 15970 requests in total, the decomposed algorithm times out only on 158 inputs, with a timeout of 15 seconds. We also analyze the size of the generated non-recursive datalog rewritings, with extremely encouraging results. Our experiments show that the decomposed algorithm performs very well on inputs from practical applications. They also confirm our initial belief that $\mathcal{EL}$-ontologies from practical applications often admit FO-rewritings. In particular, only 31 of 15970 queries turn out to not be FO-rewritable.

Throughout the paper, proof details are deferred to the appendix.

[3] For approaches to FO-rewritability for non-Horn DLs we refer the reader to [8, 5].

2 Preliminaries

We use $\mathsf{N_C}$ and $\mathsf{N_R}$ to denote countably infinite sets of concept names and role names, respectively. An $\mathcal{EL}$-*concept* is formed according to the syntax rule $C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$, a *concept inclusion (CI)* takes the form $C \sqsubseteq D$ with C and D $\mathcal{EL}$-concepts, and a *TBox* is a finite set of CIs. The semantics of concepts and TBoxes is defined as usual. We write $\mathcal{T} \models C \sqsubseteq D$ when C is subsumed by D under $\mathcal{T}$.

An ABox $\mathcal{A}$ is a set of *assertions* of the form $A(a)$ and $r(a, b)$ with A a concept name, r a role name, and a, b individual names from a countably infinite set $\mathsf{N_I}$. We use $\mathsf{ind}(\mathcal{A})$ to denote the set of individual names that occur in $\mathcal{A}$. A *concept query* is an expression $C(x)$ with C an $\mathcal{EL}$-concept and x a variable. We write $\mathcal{A}, \mathcal{T} \models C(a)$ if a is a certain answer to C given the ABox $\mathcal{A}$ and TBox $\mathcal{T}$. For an FO-query $q(x)$ with one free variable, we write $\mathcal{A} \models q(a)$ if $\mathcal{A}$ (viewed as a structure) satisfies q under the assignment that maps x to a.

A concept query $C(x)$ is *FO-rewritable under a TBox* $\mathcal{T}$ if there is an FO-formula $\varphi(x)$ such that for all ABoxes $\mathcal{A}$ and individuals a, we have $\mathcal{A}, \mathcal{T} \models C(a)$ iff $\mathcal{A} \models \varphi(a)$. In this case, we call $\varphi(x)$ an *FO-rewriting* of $C(x)$ under $\mathcal{T}$. An FO-formula $\varphi(x)$ is a *partial FO-rewriting* of A under $\mathcal{T}$ if for all ABoxes $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, $\mathcal{A} \models \varphi(a)$ implies $\mathcal{A}, \mathcal{T} \models A(a)$. Thus a partial FO-rewriting is an FO-rewriting that is sound, but not necessarily complete. When studying FO-rewritability of concept queries $C(x)$, we can assume w.l.o.g. that C is a concept name since $C(x)$ is FO-rewritable under a TBox $\mathcal{T}$ iff $A(x)$ is FO-rewritable under $\mathcal{T} \cup \{C \sqsubseteq A\}$ where A is a fresh concept name [7].

We will also consider more specific forms of FO-rewritings, in particular *UCQ-rewritings* and *non-recursive datalog rewritings*. Although, strictly speaking, non-recursive datalog rewritings are not FO-rewritings, the existence of either kind of rewriting coincides with the existence of an FO-rewriting. In particular, non-recursive datalog rewritings can be viewed as a compact representation of a UCQ-rewriting that implements structure sharing. By a *monadic datalog rewriting*, we mean a datalog rewriting in which all intensional (IDB) predicates are unary.

For our technical constructions, it will be convenient to view $\mathcal{EL}$-concepts as CQs that take the form of a directed tree. We will represent such queries as sets of atoms of the form $A(x)$ and $r(x, y)$ with A a concept name, r a role name and x, y variables, not distinguishing between answer variables and quantified variables. Tree-shapedness of a conjunctive query q then means that the directed graph $(V, \{(x, y) \mid r(x, y) \in q\})$ is a tree, where V is the set of variables in q, and that $r(x, y), s(x, y) \in q$ implies $r = s$. In the following, we will not distinguish explicitly between an $\mathcal{EL}$-concept C and its query representation. We thus use $\mathsf{var}(C)$ to denote the set of variables that occur in C and x_ε to denote the root variable in C. For an $x \in \mathsf{var}(C)$, we use $C|_x$ to denote the $\mathcal{EL}$-concept represented by (the subtree rooted at) x. When we speak of a *top-level conjunct (tlc)* of an $\mathcal{EL}$-concept C, we mean a concept name A such that $A(x_\varepsilon) \in C$ or a concept $\exists r.D$ such that $r(x_\varepsilon, y) \in C$ and $D = C|_y$. We use $\mathsf{tlc}(C)$ to denote the

set of all top-level conjuncts of C. For any syntactic object (such as a concept or a TBox), we define its *size* to be the number of symbols used to write it.

3 A Backwards Chaining Algorithm

Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and A_0 a concept name for which an FO-rewriting is to be constructed. The algorithm presented in this section constructs a set of partial rewritings of A_0 under $\mathcal{T}$ by starting from $\{A_0\}$ and then exhaustively applying the concept inclusions in $\mathcal{T}$ as rules in a backwards chaining manner. Let C and D be $\mathcal{EL}$-concepts and $\varphi = E \sqsubseteq F$ an $\mathcal{EL}$-concept inclusion. Further, let $x \in \mathsf{var}(C)$ and let there be at least one tlc G of $C|_x$ with $\models F \sqsubseteq G$. Then D is *obtained from C by applying φ at x* if D can be obtained from C by

- removing $A(x)$ for all concept names A with $\models F \sqsubseteq A$;
- removing the subtree rooted at y whenever $r(x, y) \in C$ and $\models F \sqsubseteq \exists r.(C|_y)$;
- adding $A(x)$ for all concept name A that are a tlc of E;
- adding the subtree $\exists r.H$ to x for each $\exists r.H$ that is a tlc of E.

When the exact identity of x is not important, we say that D is *obtained from C by applying φ*. This corresponds to a backwards chaining step based on so-called piece unifiers in [4, 9]. The following is immediate.

Lemma 1. *If $\mathcal{T} \models C \sqsubseteq A_0$ and D can be obtained from C by applying some CI in $\mathcal{T}$, then $\mathcal{T} \models D \sqsubseteq A_0$.*

Apart from applying CIs from the TBox as rules, our algorithm will also minimize the generated partial rewritings to attain completeness and termination. To make this precise, we introduce some notation. For $\mathcal{EL}$-concepts C and D, we write $C \preceq D$ if there is $x \in \mathsf{var}(D)$ such that $C = D \setminus D|_x$, that is the concept C is obtained from D by dropping the subtree $D|_x$ from D. We use $\preceq^*$ to denote the transitive closure of $\preceq$ and say that C *is $\preceq$-minimal with $\mathcal{T} \models C \sqsubseteq A_0$* if $\mathcal{T} \models C \sqsubseteq A_0$ and there is no $C' \preceq C$ with $\mathcal{T} \models C' \sqsubseteq A_0$. Note that if $\mathcal{T} \models C \sqsubseteq A_0$, then it is possible to find in polynomial time a $C' \preceq^* C$ that is minimal with $\mathcal{T} \models C' \sqsubseteq A_0$ (since subsumption in $\mathcal{EL}$ can be decided in PTIME).

The constructions in [7] suggest that, to achieve termination, we can use a certain form of blocking, similar to the blocking used in DL tableau algorithms. Let $\mathsf{sub}(\mathcal{T})$ denote the set of subconcepts of (concepts that occur in) $\mathcal{T}$. For each $\mathcal{EL}$-concept C and $x \in \mathsf{var}(C)$, we set $\mathsf{con}^C_{\mathcal{T}}(x) := \{D \in \mathsf{sub}(\mathcal{T}) \mid \mathcal{T} \models C|_x \sqsubseteq D\}$. We say that C is *blocked* if there are $x_1, x_2, x_3 \in \mathsf{var}(C)$ such that

1. x_1 is an ancestor of x_2, which is an ancestor of x_3 and
2. $\mathsf{con}^C_{\mathcal{T}}(x_1) = \mathsf{con}^C_{\mathcal{T}}(x_2)$ and $\mathsf{con}^{C \setminus C|_{x_3}}_{\mathcal{T}}(x_1) = \mathsf{con}^{C \setminus C|_{x_3}}_{\mathcal{T}}(x_2)$.

The algorithm is formulated in Figure 1. Note that, by Lemma 1, the concept D considered in the condition of the while loop satisfies $\mathcal{T} \models D \sqsubseteq A_0$. We are thus guaranteed to find the desired D' inside the while loop. Also note that there are potentially many different D' that we could use, and each of them will work.

```
procedure find-rewriting(A_0(x), T)
    M := {A_0}
    while there is a C ∈ M and a concept D such that
            1. D can be obtained from C by applying some CI in T and
            2. there is no D' ⪯ D with D' ∈ M then
        find a D' ⪯* D that is minimal with T ⊨ D' ⊑ A_0
        if D' is blocked then
            return 'not FO-rewritable'
        add D' to M
    return the UCQ ⋁ M.
```

Fig. 1. The backwards chaining algorithm

Example 1. Let $\mathcal{T} = \{\exists r.(B_1 \sqcap B_2) \sqsubseteq A_0,\ \exists s.B_2 \sqsubseteq B_2,\ B_1 \sqsubseteq B_2\}$. Starting with $M = \{A_0\}$ and applying the first CI to $C = A_0$, we get $M = \{A_0, \exists r.(B_1 \sqcap B_2)\}$. Applying the second CI to $C = \exists r.(B_1 \sqcap B_2)$ then yields $D = \exists r.(B_1 \sqcap \exists s.B_2)$ which is not minimal with $\mathcal{T} \models D \sqsubseteq A_0$ as witnessed by $D' = \exists r.B_1$, which is added to M. At this point, rule application stops and the UCQ $\bigvee M$ is returned.

It is illustrative to try Example 1 without applying the minimization step. We then find a blocked concept in M, which shows that without minimization the algorithm is incomplete. It can also be seen that dropping minimization results in non-termination since the out-degree of concepts in M can grow unboundedly.

We now establish correctness and termination, showing first that, if the algorithm claims to have found an FO-rewriting, then this is indeed the case.

Proposition 1 (Soundness). *If the algorithm returns $\bigvee M$, then $\bigvee M$ is an FO-rewriting of A_0 under $\mathcal{T}$.*

Proof.(sketch) Let $\mathcal{A}$ be an ABox. We have to show: (1) if $\mathcal{A} \models \bigvee M(a_0)$, then $\mathcal{A}, \mathcal{T} \models A_0(a_0)$; (2) if $\mathcal{A}, \mathcal{T} \models A_0(a_0)$, then $\mathcal{A} \models \bigvee M(a_0)$. For Point 1, assume that $\mathcal{A} \models \bigvee M(a_0)$. Then there is a $C \in M$ with $\mathcal{A} \models C(a_0)$. Consequently $\mathcal{A}, \mathcal{T} \models C(a_0)$. By construction of M, all its elements C satisfy $\mathcal{T} \models C \sqsubseteq A_0$, thus $\mathcal{A}, \mathcal{T} \models A_0(a_0)$ as required.

For Point 2, we essentially follow the proof strategy from [4], based on the chase procedure. If $\mathcal{A}, \mathcal{T} \models A_0(a_0)$, then $A_0(a_0) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$ and consequently, there is a sequence of ABoxes $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k$ that *demonstrates* $A_0(a) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, that is, each $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by a single chase step and $A_0(a) \in \mathcal{A}_k$. It thus suffices to prove by induction on k that if $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ is a chase sequence that demonstrates $A_0(a_0) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, then $\mathcal{A} \models \bigvee M(a_0)$. This is slightly tedious, but straightforward. ❑

Note that the generated UCQ-rewritings are not necessarily of minimal size. It is possible to attain minimal-size rewritings by using a stronger form of minimality when constructing the concept D', namely by redefining the relation "$\preceq$" so that $C \preceq D$ if there is a root-preserving homomorphism from C to D (c.f. the notion of *most-general rewritings* in [9]). As a consequence, the $\preceq$-minimal concept D'

with $\mathcal{T} \models D' \sqsubseteq A$ can then be of size exponential in the size of D. However, D' can still be constructed in output-polynomial time.

Proposition 2 (Completeness). *If the algorithm returns 'not FO-rewritable', then A_0 has no FO-rewriting under $\mathcal{T}$.*

Proof. By Theorem 2 in [6], it suffices to show that if the algorithm returns 'not FO-rewritable', then

($*$) for every $k > 0$, there is a concept C whose depth exceeds k and such that $\mathcal{T} \models C \sqsubseteq A_0$ and $\mathcal{T} \not\models C|_k^- \sqsubseteq A_0$,

where $C|_k^-$ denotes the concept obtained from C be removing all variables whose depth exceeds k. Using a pumping argument, one can show the following sufficient condition for ($*$).

Fact. If there is a concept C that is blocked with variables $x_1, x_2, x_3 \in \mathsf{var}(C)$, $\mathcal{T} \models C \sqsubseteq A_0$, and $\mathcal{T} \not\models C \setminus C|_{x_3} \sqsubseteq A_0$, then ($*$) holds.

Now assume the algorithm returns 'not FO-rewritable'. Then there is a concept D that is minimal with $\mathcal{T} \models D \sqsubseteq A_0$ and that is blocked with variables x_1, x_2, x_3. By minimality of D, $\mathcal{T} \not\models (D \setminus D|_{x_3}) \sqsubseteq A_0$ and ($*$) follows. ❑

We prove in the appendix that the algorithm always terminates by showing that all concepts in M have outdegree at most n and depth at most 2^{2n}, n the size of $\mathcal{T}$. As remarked in [7], the size of UCQ-rewritings can be triple exponential in the size of $\mathcal{T}$, and thus the same is true for the runtime of the presented algorithm. While this worst case is probably not encountered in practice, the size of M can become prohibitively large for realistic inputs. For this reason, we propose an improved algorithm in the subsequent section, which produces non-recursive datalog rewritings instead of UCQ-rewritings and whose runtime is at most single exponential.

4 A Decomposed Algorithm

The algorithm presented in this section consists of three phases. In the first phase, a certain set Γ is computed that can be viewed as a decomposed representation of the set M from Section 3 in the sense that we store only single nodes of the tree-shaped concepts in M, rather than entire concepts. In the second phase, we compute a certain set Ω that enriches the node representation provided by Γ with sets of logical consequences as mentioned in Point 2 of the definition of *blocked* concepts. In the third phase, we first execute a certain cycle check on Ω, which corresponds to checking the existence of a blocked concept in M. If no cycle is found, we can read off a rewriting from Ω.

Assume that $\mathcal{T}$ is a TBox and A_0 a concept name for which we want to compute an FO-rewriting. To present the algorithm, it is convenient to decompose conjunctions on the right-hand side of CIs, that is, to assume that $\mathcal{T}$ consists of CIs of the form $C \sqsubseteq A$, A a concept name, or $C \sqsubseteq \exists r.D$. We start with describing the construction of Γ, whose elements we call node pairs. A *node pair* has

the form (C, S), where $C \in \mathsf{sub}(\mathcal{T})$ and $S \subseteq \mathsf{sub}(\mathcal{T})$ is a set of concept names and concepts of the form $\exists r.C$. Intuitively, a node pair (C, S) describes a set of concepts D (subtrees of concepts in Γ) such that $\mathcal{T} \models D \sqsubseteq C$ and the following conditions are satisfied:

(i) if $A \in S$ for a concept name A, then A is a tlc of D;
(ii) if $\exists r.E \in S$, then there is a tlc $\exists r.E'$ of D such that $\mathcal{T} \models E' \sqsubseteq E$.

The computation of Γ starts with $\{(A_0, \{A_0\})\}$ and proceeds by exhaustively applying the following two rules:

(r1) if $(C, S) \in \Gamma$, $D \sqsubseteq A \in \mathcal{T}$ and $A \in S$, then add pair $(C, (S \setminus \{A\}) \cup \mathsf{tlc}(D))$
(r2) if $(C, S) \in \Gamma$, $D \sqsubseteq \exists r.F \in \mathcal{T}$, and there is an $\exists r.G \in S$ with $\mathcal{T} \models F \sqsubseteq G$, then add the pair $(C, (S \setminus \{\exists r.G \mid \mathcal{T} \models F \sqsubseteq G\}) \cup \mathsf{tlc}(D))$

After applying either rule, we also have to add the pair $(G, \mathsf{tlc}(G))$ for every $\exists r.G \in \mathsf{sub}(D)$ to trigger further derivation.

The set Γ represents a (potentially infinitary) UCQ-rewriting of A_0 under $\mathcal{T}$ in a sense that we make precise now. Let $\widehat{\Gamma}$ be the set obtained as the limit of the sequence of sets $\widehat{\Gamma}_0, \widehat{\Gamma}_1, \ldots$ defined as follows:

- $\widehat{\Gamma}_0 := \{(C, \bigsqcap S) \mid (C, S) \in \Gamma \text{ and } S \subseteq \mathsf{N_C}\}$.
- $\widehat{\Gamma}_{i+1}$ is $\widehat{\Gamma}_i$ extended with all pairs (C, D) such that there are $(C, S) \in \Gamma$ and $(G, C_{r,G}) \in \widehat{\Gamma}_i$ for each $\exists r.G \in S$ and $D = \bigsqcap_{A \in S \cap \mathsf{N_C}} A \sqcap \bigsqcap_{\exists r.G \in S} \exists r.C_{r,G}$.

Proposition 3 (Soundness and Completeness of $\widehat{\Gamma}$). *For all ABoxes $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, we have $\mathcal{A}, \mathcal{T} \models A_0(a)$ iff there is a $(A_0, D) \in \widehat{\Gamma}$ with $\mathcal{A} \models D(a)$.*

Note that Γ provides us with a sufficient condition for FO-rewritability and suggests a way to produce a non-recursive datalog rewriting. In fact, if Γ is *acyclic* in the sense that the directed graph

$$G_\Gamma = (\Gamma, \{((C, S), (C', S')) \mid S \text{ contains a concept } \exists r.C'\})$$

contains no cycle, then $\widehat{\Gamma}$ is finite and we obtain a non-recursive datalog program Π_Γ that is a rewriting of A_0 under $\mathcal{T}$ by taking the rule

$$P_C(x) \leftarrow \bigwedge_{A \in S} A(x) \wedge \bigwedge_{\exists r.D \in S} (\, r(x, y_{r,D}) \wedge P_D(y_{r,D}) \,)$$

for each $(C, S) \in \Gamma$ and using A_0 as the goal predicate. However, if Γ is not acyclic, then A_0 could still be FO-rewritable under $\mathcal{T}$, but the above program will be recursive. To deal with this problem, we need the next two phases.

We construct a set of node tuples Ω from Γ by further annotating and duplicating the pairs in Γ. A *node tuple* takes the form $t = (C_t, S_t, \mathsf{con}_t, s_t, \mathsf{xcon}_t)$ where C_t and S_t have the same form as the components of node pairs in Γ, $\mathsf{con}_t \subseteq \mathsf{sub}(\mathcal{T})$, s_t is an existential restriction in S_t or the special symbol "$-$", and xcon_t is either a subset of $\mathsf{sub}(\mathcal{T})$ or the special symbol "$-$". Intuitively, a node tuple $t \in \Omega$ describes a set of concepts D (subtrees of concepts in Γ) such that (C_t, S_t) describes D in the way described above and the following additional conditions are satisfied:

(iii) for each $E \in \mathsf{sub}(\mathcal{T})$, we have $\mathcal{T} \models D \sqsubseteq E$ iff $E \in \mathsf{con}_t$;
(iv) in the subtree of D rooted at s_t there is a leaf node such that for the concept D' obtained by dropped this node and each $E \in \mathsf{sub}(\mathcal{T})$, we have $\mathcal{T} \models D' \sqsubseteq E$ iff $E \in \mathsf{xcon}_t$.

When S_t contains no existential restrictions, we use "$-$" in the last two components. To understand s_t, it is useful to think of D as a tree and of s_t as a selected successor of the root of that tree. We start the construction of Ω with

$$\Omega_0 := \{(C, S, \mathsf{con}_\mathcal{T}(S), -, -) \mid (C, S) \in \Gamma \text{ with } S \subseteq \mathsf{N_C}\},$$

where for a set of concepts M, $\mathsf{con}_\mathcal{T}(M)$ denotes the set of concepts $D \in \mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models \bigsqcap M \sqsubseteq D$. We call the tuples in Ω_0 *leaf tuples*. The final set Ω is constructed by exhaustively applying the following rule:

(r3) If t is a node tuple such that $\exists r_0.D_0, \ldots, \exists r_n.D_n$ are the existential restrictions in S_t, $s_t = \exists r_\ell.D_\ell$, and $t_0, \ldots, t_n \in \Omega$ with $C_{t_i} = D_i$ for $0 \leq i \leq n$, then add t to Ω if the following conditions are satisfied:
- there is a node pair $(C_t, S) \in \Gamma$ with $S_t \subseteq S$ and $S \cap \mathsf{N_C} = S_t \cap \mathsf{N_C}$;
- $\mathsf{con}_t = \mathsf{con}_\mathcal{T}(M)$, where $M = \bigcup(S_t \cap \mathsf{N_C}) \cup \{\exists r_i.G \mid i \leq n \text{ and } G \in \mathsf{con}_{t_i}\}$;
- $\mathsf{xcon}_t = \mathsf{con}_\mathcal{T}(M')$, where M' is

$$\bigcup(S_t \cap \mathsf{N_C}) \cup \{\exists r_\ell.G \mid G \in \mathsf{xcon}_{t_\ell}\} \cup \{\exists r_i.G \mid \ell \neq i \leq n \text{ and } G \in \mathsf{con}_{t_i}\}.$$

In Points 3, the concept $\exists r.-$ (which might occur in the set M') is identified with $\top$. For $t, t' \in \Omega$, we write $t \rightsquigarrow_\Omega t'$ if there are $t_0, \ldots, t_n \in \Omega$ that satisfy the conditions listed in (r3) and such that $t' = t_\ell$, that is, t' is the tuple that was chosen for the selected successor. Note that the computation of the sets con_t and xcon_t is complete, relying on the following observation [13].

Lemma 2. *For any concept $C = A_1 \sqcap \cdots \sqcap A_n \sqcap \exists r_1.G_1 \sqcap \cdots \sqcap \exists r_m.G_m$,*

$$\mathsf{con}_\mathcal{T}(C) = \mathsf{con}_\mathcal{T}(\{A_1, \ldots, A_n\} \cup \bigcup_{1 \leq i \leq m} \{\exists r_i.D \mid D \in \mathsf{con}_\mathcal{T}(G_i)\})$$

We now describe the third and last phase of the algorithm, which first checks whether an FO-rewriting exists at all and, if so, produces a rewriting that takes the form of a non-recursive monadic datalog program.

We start with introducing the relevant notion of a cycle. A tuple $t \in \Omega$ is a *root tuple* if $A_0 \in \mathsf{con}_t$ and $A_0 \notin \mathsf{xcon}_t$. A *path through* Ω is a finite sequence of node tuples $t_1, \ldots, t_k$ from Ω such that $t_i \rightsquigarrow_\Omega t_{i+1}$ for $1 \leq i < k$. A tuple $t \in \Omega$ is *looping* if there is a path $t_1, \ldots, t_k$ through Ω of length at least one such that $t = t_1$, $\mathsf{con}_t = \mathsf{con}_{t_k}$, and $\mathsf{xcon}_t = \mathsf{xcon}_{t_k}$. We say that Ω *contains a root cycle* if there are tuples $t, t' \in \Omega$ such that t is a root tuple, t' is a looping tuple, and t' is reachable from t along $\rightsquigarrow_\Omega$.

Proposition 4. *A_0 is not FO-rewritable under $\mathcal{T}$ if and only if Ω contains a root cycle.*

Proof.(sketch) "if". Assume that Ω contains a root cycle. Using this cycle as a guide, we show how to find a concept C that is blocked in the sense of Section 3 and satisfies $\mathcal{T} \models C \sqsubseteq A_0$ as well as $\mathcal{T} \not\models (C \setminus C|_{x_3}) \sqsubseteq A_0$, where x_3 is as in the definition of 'blocked'. Once we have constructed such a concept C, we can again rely on the results of [6], as in the proof of Proposition 2.

"only if". Assuming that Ω contains no root cycle, we show below how to constuct a non-recursive datalog-rewriting of A_0 under $\mathcal{T}$, thus A_0 is FO-rewritable under $\mathcal{T}$. ❑

As suggested by Proposition 4, our algorithm first checks whether Ω contains a root cycle and, if so, returns 'not FO-rewritable'. Otherwise, it constructs a non-recursive datalog program $\Pi_{\mathcal{T},A_0}$ as follows. First, we drop from Ω all tuples that are not reachable from a root tuple along an $\rightsquigarrow_\Omega$-path. For $t, t' \in \Omega$ and $\exists r.D \in S_t$, we write $t \rightsquigarrow_{\exists r.D} t'$ if there is a tuple $\widehat{t} = (C_t, S_t, \mathsf{con}_t, \exists r.D, \mathsf{xcon}_{\widehat{t}}) \in \Omega$ such that $\widehat{t} \rightsquigarrow_\Omega t'$. Note that, by definition of "$\rightsquigarrow_\Omega$", $t \rightsquigarrow_{\exists r.D} t'$ implies $C_{t'} = D$. Now, $\Pi_{\mathcal{T},A_0}$ contains for every $t \in \Omega$, the rule

$$P_{C_t,\mathsf{con}_t}(x) \leftarrow \bigwedge_{A \in S_t \cap \mathsf{N_C}} A(x) \wedge \bigwedge_{\exists r.D \in S_t} (r(x, y_{r,D}) \wedge \bigvee_{t' \in \Omega | t \rightsquigarrow_{\exists r.D} t'} P_{D,\mathsf{con}_{t'}}(y_{r,D}))$$

Note that the disjunctions can be removed by introducing auxiliary IDB predicates, without causing a significant blowup. The goal predicates of $\Pi_{\mathcal{T},A_0}$ are all predicates of the form $P_{A_0,\mathsf{con}}(x)$ with $A_0 \in \mathsf{con}$.

Theorem 1.
1. *The program $\Pi_{\mathcal{T},A_0}$ is a rewriting of A_0 under $\mathcal{T}$.*
2. *If Ω contains no root cycle, then $\Pi_{\mathcal{T},A_0}$ is non-recursive.*

Even if Ω contains no root-cycles, the program $\Pi_{\mathcal{T},A_0}$ may have up to (single) exponentially many IDB predicates. We observe that this cannot be significantly improved without giving up monadicity.

Theorem 2. *There is a family of TBoxes $\mathcal{T}_1, \mathcal{T}_2, \ldots$ such that for all $n \geq 1$, $\mathcal{T}_n$ is of size $\mathcal{O}(n^2)$, the concept name A_0 is FO-rewritable under $\mathcal{T}_n$, and the smallest non-recursive monadic datalog rewriting has size at least 2^n.*

Let us briefly analyze the complexity of the decomposed algorithm. It is easy to verify that the number of Γ-pairs and Ω-tuples is singly exponential in the size of $\mathcal{T}$ and that all required operations for building Γ and Ω and for determining the existence of a root cycle require only polynomial time. By Proposition 4, we have thus found an ExpTime algorithm for deciding FO-rewritability of $\mathcal{EL}$-concept queries. This is almost optimal as the problem we are dealing with is PSpace-complete and becomes ExpTime-hard if slightly varied, see [7].

5 Experiments

We have implemented the decomposed algorithm in Java and conducted a number of experiments. The implementation is not highly optimized, but some aspects of handling the set Γ are worth to point out. In particular, we use numbers

TBox	concept inclusions	concept names	role names	timeouts	not FO-rewritable
XP	1046	906	27	0	1
NBO	1468	962	8	0	6
ENVO	1848	1538	7	0	7
FBbi	567	517	1	0	0
MOHSE	3665	2203	71	2	1
not-galen	4636	2748	159	149	0
SO	2740	1894	13	7	16

Table 1. TBoxes used in the experiments.

to represent subconcepts in $\mathcal{T}$, store the S-component of each pair $(C, S) \in \Gamma$, which is a set of subconcepts of $\mathcal{T}$, as an ordered set, and use so-called *tries* as a data structure to store Γ. We remove pairs $(C, S) \in \Gamma$ where S is not minimal, that is, for which there is a $(C, S') \in \Gamma$ with $S' \subsetneq S$. It is easy to see that this optimization does not compromise correctness.

The experiments were carried out on a Linux (3.2.0) machine with a 3.5Ghz quad-core processor and 8GB of RAM. Although a large number of $\mathcal{EL}$-TBoxes is available on the web and from various repositories, most of them are *acyclic TBoxes* in the traditional DL sense, that is, the left-hand sides of all CIs are concept names, there are no two CIs with the same left-hand side, and there are no syntactic cycles. Since concept queries are always FO-rewritable under acyclic $\mathcal{EL}$-TBoxes [6], such TBoxes are not useful for our experiments. We have identified seven TBoxes that do not fall into this class, listed in Table 1 together with the number of concept inclusions, concept names, and role names that they contain. All TBoxes together with information about their origin are available at http://tinyurl.com/q96q34z.

For each of these TBoxes, we have applied the decomposed algorithm to every concept name in the TBox. In some rare cases, the set Γ has reached excessive size, resulting in non-termination. We have thus established a 15 second timeout for the Γ-phase of the algorithm. With that timeout, our algorithm was able to decide FO-rewritability in almost all of the cases, see Table 1. The overall runtime for all our experiments as a batch job (15970 invocations of the algorithm) took only 80 minutes. The generated non-recursive datalog-rewritings are typically of very reasonable size. The number of rules in the rewriting is displayed in the upper part of Figure 2; for example, for NBO, about 55% of all rewritings consist of a single rule, about 18% have two or three rules, about 10% have 4–7 rules, and so on. Note that the x-axis has logarithmic scale. The size of the rule bodies is typically very small, between one and two atoms in the vast majority of cases, and we have never encountered a rule with more than ten body atoms. It is also interesting to consider the number of IDB predicates in a rewriting, as intuitively these correspond to views that have to be generated by a database system that executes the query. As shown in the lower part of Figure 2, the number of IDB predicates is rather small, and considerably lower than the number of rules in the produced programs (we again use logarithmic scale on the x-axis).

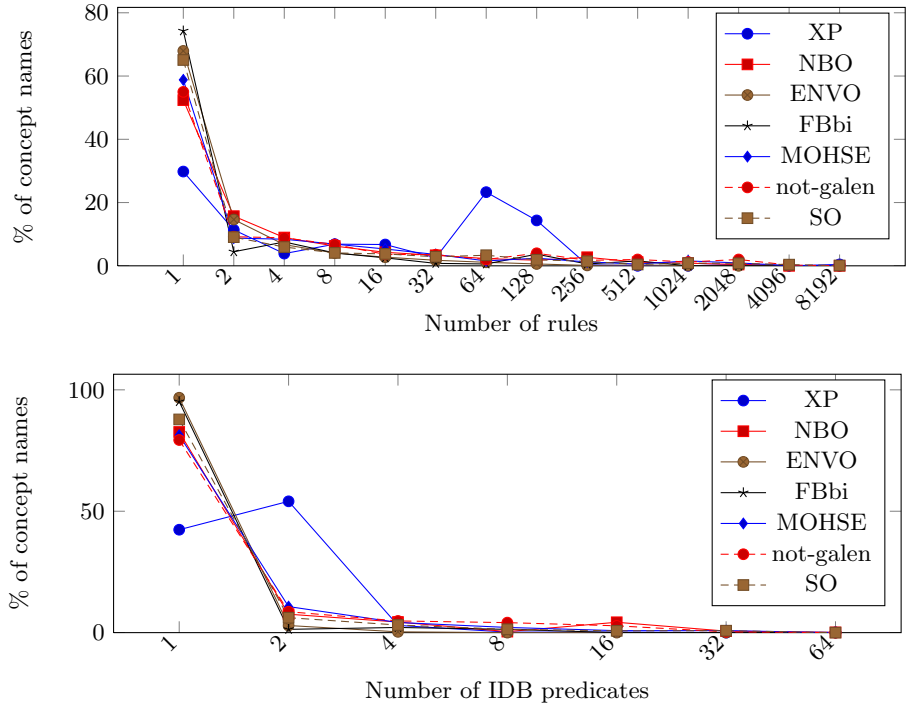

Fig. 2. Number of rules and IDB predicates in the rewriting

The experiments also confirm our initial belief that ontologies which are used in practical applications have a simple structure. As shown in Table 1, the number of concept names that are not FO-rewritable is extremely small. Moreover, if a concept name was FO-rewritable, then we were always able to determine this already in the Γ-phase of our algorithm, without ever entering the Ω-phase. Note, though, that for those cases that turned out to be not FO-rewritable, we had to go through the full Ω-construction.

6 Outlook

We plan to optimize the implementation of the decomposed approach further to eliminate the timeouts we encountered in the experiments. Moreover, we plan to extend the algorithm and implementation in several ways. First, we plan to generalize the approach from concept queries to conjunctive queries. Second, in many applications the ABox signature (i.e., the concept and role names occurring in the ABoxes) is a small subset of the signature of the TBox [2, 7]. Queries that are not FO-rewritable without any restriction on the ABox signature can become FO-rewritable under smaller ABox signatures. We therefore plan to extend the decomposed approach to arbitrary ABox signatures. Finally, we plan to extend our approach to more expressive Horn-DLs such as $\mathcal{ELI}$ with role inclusions and thereby unify DL-Lite and $\mathcal{EL}$ query rewriting approaches in one framework.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baader, F., Bienvenu, M., Lutz, C., Wolter, F.: Query and predicate emptiness in description logics. In: KR. pp. 192–202 (2010)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: IJCAI. pp. 364–369 (2005)
4. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. 175(9-10), 1620–1654 (2011)
5. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: a study through Disjunctive Datalog, CSP, and MMSNP. In: PODS. pp. 213–224 (2013)
6. Bienvenu, M., Lutz, C., Wolter, F.: Deciding FO-rewritability in $\mathcal{EL}$. In: Description Logics. pp. 70–80 (2012)
7. Bienvenu, M., Lutz, C., Wolter, F.: First order-rewritability of atomic queries in horn description logics. In: IJCAI. pp. 754–760 (2013)
8. Kaminski, M., Grau, B.C.: Sufficient conditions for first-order and datalog rewritability in $\mathcal{ELU}$. In: Description Logics. pp. 271–293 (2013)
9. König, M., Leclère, M., Mugnier, M.L., Thomazo, M.: A sound and complete backward chaining algorithm for existential rules. In: RR. pp. 122–138 (2012)
10. Kontchakov, R., Rodriguez-Muro, M., Zakharyaschev, M.: Ontology-based data access with databases: A short course. In: Reasoning Web. pp. 194–229 (2013)
11. Lutz, C., İnanç Seylan, Toman, D., Wolter, F.: The combined approach to OBDA: Taming role hierarchies using filters. In: ISWC. pp. 314–330 (2013)
12. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic el using a relational database system. In: IJCAI. pp. 2070–2075 (2009)
13. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. J. Symb. Comput. pp. 194–228 (2010)
14. Mora, J., Corcho, Ó.: Engineering optimisations in query rewriting for OBDA. In: I-SEMANTICS. pp. 41–48 (2013)
15. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic 8(2), 186–209 (2010)
16. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
17. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Description Logics. pp. 451–458 (2007)
18. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR. pp. 290–300 (2010)

A Proofs for Section 3

For the proof of Proposition 1, we remind the reader of the standard chase procedure. The chase is a forward chaining procedure that exhaustively applies the CIs of a TBox to an ABox in a rule-like fashion. Its final result is a (potentially infinite) ABox in which all consequences of $\mathcal{T}$ are materialised. To describe the procedure in detail and in the following proof it is helpful to regard $\mathcal{EL}$-concepts C as tree-shaped ABoxes $\mathcal{A}_C$. $\mathcal{A}_C$ can be obtained from the concept query corresponding to C by identifying its individual variables with individual names. Now let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $\mathcal{A}$ an ABox. A *chase step* consists in choosing a CI $C \sqsubseteq D \in \mathcal{T}$ and an individual $a \in \mathsf{ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{T} \models C(a)$, and then extending $\mathcal{A}$ by taking a copy $\mathcal{A}_D$ of D viewed as an ABox with root a and such that all non-roots are fresh individuals, and then setting $\mathcal{A} := \mathcal{A} \cup \mathcal{A}_D$. The *result of chasing $\mathcal{A}$ with $\mathcal{T}$* , denoted with $\mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, is the ABox obtained by exhaustively applying chase steps to $\mathcal{A}$ in a fair way. It is standard to show that for all $\mathcal{EL}$-concepts C, we have $\mathcal{A}, \mathcal{T} \models C(a)$ iff $\mathsf{chase}_{\mathcal{T}}(\mathcal{A}) \models C(a)$.

Proposition 1 (Soundness). If the algorithm returns $\bigvee M$, then $\bigvee M$ is an FO-rewriting of A_0 under $\mathcal{T}$.

Proof. Let $\mathcal{A}$ be an ABox. We have to show: (i)

1. if $\mathcal{A} \models \bigvee M(a_0)$, then $\mathcal{A}, \mathcal{T} \models A_0(a_0)$;
2. if $\mathcal{A}, \mathcal{T} \models A_0(a_0)$, then $\mathcal{A} \models \bigvee M(a_0)$.

We start with Point 1. Thus assume that $\mathcal{A} \models \bigvee M(a_0)$. Then there is a $C \in M$ with $\mathcal{A} \models C(a_0)$. By construction of M, we have $\mathcal{T} \models C \sqsubseteq A_0$, thus $\mathcal{A}, \mathcal{T} \models A_0(a_0)$ as required.

For Point 2, assume that $\mathcal{A}, \mathcal{T} \models A_0(a_0)$. Then $A_0(a_0) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$. Consequently, there is a sequence of ABoxes $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k$ that *demonstrates* $A_0(a) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, that is, each $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by a single chase step and $A_0(a) \in \mathcal{A}_k$. It thus suffices to prove by induction on k that

($*$) if $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ is a chase sequence that demonstrates $A_0(a_0) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, then $\mathcal{A} \models \bigvee M(a_0)$.

The induction start is trivial: for $k = 0$, $A_0(a_0) \in \mathcal{A}_k$ implies $A_0(a_0) \in \mathcal{A}$. Since $A_0 \in M$, we have $\mathcal{A} \models \bigvee M(a_0)$. For the induction step, assume that $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ is a chase sequence that demonstrates $A_0(a) \in \mathsf{chase}_{\mathcal{T}}(\mathcal{A})$, with $k > 0$. Applying IH to the subsequence $\mathcal{A}_1, \ldots, \mathcal{A}_k$, we obtain that $\mathcal{A}_1 \models \bigvee M(a_0)$. Thus there is a $C \in M$ with $\mathcal{A}_1 \models C(a_0)$. Assume that $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by choosing $E \sqsubseteq F \in \mathcal{T}$ and $a \in \mathsf{Ind}(\mathcal{A}_0)$ with $\mathcal{A}_0 \models E(a)$, and adding a copy of the ABox $\mathcal{A}_F$ to $\mathcal{A}_0$:

- $A(a)$ for each concept name A that occurs as a top-level conjunct in F;
- the sub-ABox $\mathcal{A}_{\exists r.G}$ rooted at a for each $\exists r.G$ that is a top-level conjunct of F.

We might or might not have $\mathcal{A}_0 \models C(a_0)$, depending on whether or not the truth of C at a_0 depends on the additions due to applying $E \sqsubseteq F$ as a (forward) rule at a. If $\mathcal{A}_0 \models C(a_0)$ does hold, then we are done. Otherwise, let h be a homomorphism from C to $\mathcal{A}_1$ with $h(x_\varepsilon) = a_0$ and let $x_1, \ldots, x_n$ be all elements of $\mathsf{var}(C)$ such that $h(x_i) = a$ and there is at least one tlc G_i of $C|_{x_i}$ with $F \sqsubseteq G_i$. There must be at least one such x_i since, otherwise, h does not depend on any assertions added in the construction of $\mathcal{A}_1$ from $\mathcal{A}_0$ and thus witnesses $\mathcal{A}_0 \models C(a_0)$, a contradiction. Let the concepts $C_0, \ldots, C_n$ be such that $C_0 = C$ and C_{i+1} can be obtained from C_i by doing the following if $x_i \in \mathsf{var}(C_i)$ (otherwise, just set $C_{i+1} := C_i$):

1. remove $A(x_i)$ for all concept names A with $\models F \sqsubseteq A$;
2. remove the subtree rooted at y whenever $r(x_i, y) \in C$ and $\models F \sqsubseteq \exists r.(C|_y)$;
3. add $A(x_i)$ for all concept names A that are a top-level conjuncts of E;
4. add the subtree $\exists r.H$ to x_i for each $\exists r.H$ that is a top-level conjunct of E;
5. minimize the resulting C_i', that is, choose $C_{i+1} \preceq^* C_i'$ such that C_{i+1} is minimal with $\mathcal{T} \models C_{i+1} \sqsubseteq A_0$.

It is easy to prove by induction on i that $C_i \in M$ for all $i \leq n$. It thus remains to argue that $\mathcal{A}_0 \models C_n(a_0)$. To do this, we produce maps $h_0, \ldots, h_n$ such that h_i is a homomorphism from C_i to $\mathcal{A}_1$ with $h_i(x_\varepsilon) = a_0$ and such that $h_i(x_j) = a$ if $x_j \in \mathsf{var}(C_i)$, for all $i \leq n$. Start with $h_0 = h$. To produce h_{i+1} from h_i, first restrict h_i to the 'remainder' of C_i after the removals in Steps 1 and 2 were carried out. Then extend h_i to cover all fresh elements introduced via the subtrees $\exists r.H$ in Step 4. Note that, since $\mathcal{A}_0 \models E(a)$ and $h_i(x_i) = a$, this is possible. For the same reason, the resulting homomorphism h_i' respects all the concept assertions added in Step 3. Finally, to deal with the minimization in Step 5, restrict h_i' to $\mathsf{var}(C_{i+1})$.

By construction of the concepts $C_0, \ldots, C_n$ and the homomorphisms $h_0, \ldots, h_n$, there is no atom in C_n such that the image of the atom under h_n was added by applying $E \sqsubseteq F$ as a (forward) rule at a. To show this, assume to the contrary that there is such an atom $A(x)$ in C_n. There are two cases:

1. $h(x) = a$.
   Then $x = x_i$ for some i. Since $A(h(x)) = A(a)$ was added by the application of $E \sqsubseteq F$, A is a top-level conjunct of F. Consequently, $A(x)$ was removed in Step 1 when constructing C_{i+1} from C_i, in contradiction to $A(x)$ being in C_n.
2. $h(x) \neq a$.
   Then $h(x)$ is a non-root node of the sub-ABox $\mathcal{A}_{\exists r.G}$ of $\mathcal{A}_1$, where $\exists r.G$ is a top-level conjunct of F. Consider the unique path in C from x_ε to x, that is, the sequence of individuals $y_0, \ldots, y_\ell$ with $y_0 = x_\varepsilon$ and $y_\ell = x$ such that $r_i(y_i, y_{i+1}) \in C$ for some r_i, for all $i < \ell$. We find a corresponding path $h(y_0), \ldots, h(y_\ell)$ in $\mathcal{A}_1$, and since $\mathcal{A}_1$ is tree-shaped, a must be on that second path. Let y_p be such that $h(y_p) = a$. We must have $y_p = x_i$ for some i. Then $r_p = r$ and $\mathcal{A}_{\exists r.G} \models \exists r.(C|y_{p+1})(a)$. Hence $\models F \sqsubseteq \exists r.(C|y_{p+1})$ since $\exists r.G$ is a top-level conjunct of F. Consequently, the subtree of C rooted at y_{p+1} was

removed in Step 2 when constructing C_{i+1}, in contradiction to $A(x)$ being in C_n.

The case of role atoms is similar to subcase 2 above, but simpler. We have thus shown that there is no atom in C_n such that the image of this atom under h_n was added by applying $E \sqsubseteq F$ as a (forward) rule at a. Consequently $\mathcal{A}_0 \models C_n(a_0)$ via h_n. This completes the proof. ❑

Proposition 2 (Completeness). If the algorithm returns 'not FO-rewritable', then A_0 has no FO-rewriting under $\mathcal{T}$.

Proof. It remains to prove the following

Fact. If there is a concept C that is blocked with variables $x_1, x_2, x_3 \in \mathsf{var}(C)$, $\mathcal{T} \models C \sqsubseteq A_0$, and $\mathcal{T} \not\models C \setminus C|_{x_3} \sqsubseteq A_0$, then $(*)$ holds, where $(*)$ states that for every $k > 0$, there is a concept C whose depth exceeds k and such that $\mathcal{T} \models C \sqsubseteq A_0$ and $\mathcal{T} \not\models C|_k^- \sqsubseteq A_0$,

Consider a concept C that is blocked with variables $x_1, x_2, x_3 \in \mathsf{var}(C)$, $\mathcal{T} \models C \sqsubseteq A_0$, and $\mathcal{T} \not\models C \setminus C|_{x_3} \sqsubseteq A_0$. So the following conditions are satisfied:

1. x_1 is a (proper) ancestor of x_2, which is a (not necessarily proper) ancestor of x_3,
2. $\mathsf{con}_{\mathcal{T}}^{C}(x_1) = \mathsf{con}_{\mathcal{T}}^{C}(x_2)$ and $\mathsf{con}_{\mathcal{T}}^{C\setminus C|_{x_3}}(x_1) = \mathsf{con}_{\mathcal{T}}^{C\setminus C|_{x_3}}(x_2)$.

Let G be obtained from C by replacing the subconcept $C|_{x_2}$ with a copy of $C|_{x_1}$ in which every variable x has been renamed to x'. It can be proved that

1. $\mathsf{con}_{\mathcal{T}}^{C}(x_1) = \mathsf{con}_{\mathcal{T}}^{G}(x_{1'}) = \mathsf{con}_{\mathcal{T}}^{G}(x_1)$;
2. $\mathsf{con}_{\mathcal{T}}^{C\setminus C|_{x_3}}(x_1) = \mathsf{con}_{\mathcal{T}}^{G\setminus G|_{x'_3}}(x'_1) = \mathsf{con}_{\mathcal{T}}^{G\setminus G|_{x'_3}}(x_1)$;
3. $\mathsf{con}_{\mathcal{T}}^{C}(x_\varepsilon) = \mathsf{con}_{\mathcal{T}}^{G}(x_\varepsilon)$;
4. $\mathsf{con}_{\mathcal{T}}^{C\setminus C|_{x_3}}(x_\varepsilon) = \mathsf{con}_{\mathcal{T}}^{G\setminus G|_{x'_3}}(x_\varepsilon)$.

In particular, we thus have that G is blocked with variables x_1, x'_1, and x'_3, $\mathcal{T} \models G \sqsubseteq A_0$ and $\mathcal{T} \not\models G \setminus G|_{x'_3} \sqsubseteq A_0$. Also note that the depth of x'_3 in G is larger than the depth of x_3 in C. We now apply the same construction to G again until the copy of x_3 has depth exceeding k. For the resulting concept G' we have $\mathcal{T} \models G' \sqsubseteq A_0$ and $\mathcal{T} \not\models G'|_k^- \sqsubseteq A_0$, as required. ❑

Proposition 5. *The algorithm terminates on any input.*

Proof. Let $\mathcal{T}$ be of size n. Every concept of depth larger than 2^{2n} must be blocked. For this reason, M can only contain concepts of depth at most 2^{2n}. Since every round of the while loop adds a fresh concept to M, it thus suffices to show that each concept in M has outdegree at most n. This is in fact a consequence of minimization. Assume that D' is minimal with $\mathcal{T} \models D' \sqsubseteq A_0$ and that, to the contrary of what we aim to show, there is some $x \in \mathsf{var}(D')$ that has successors $r_1(x, x_1), \ldots, r_{n+1}(x, x_{n+1})$. For each existential restriction $\exists r.C$ in $\mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models D'|_x \sqsubseteq \exists r.C$, choose a successor x_i such that $r_i = r$ and

$\mathcal{T} \models D'|_{x_i} \sqsubseteq C$. Let D'' be obtained from D' by dropping all subtrees rooted at nodes x_i that were not chosen in this way. Then we have $\mathcal{T} \models D'' \sqsubseteq A_0$. Since at least one x_i-rooted subtree was dropped in the construction of D'' from D', this contradicts the minimality of D'. ❑

B Proofs for Section 4

We split the proof of Proposition 3 into a soundness and a completeness part.

Lemma 3 (Soundness of $\widehat{\Gamma}$). *For all ABoxes $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, if there is a $(A_0, D) \in \widehat{\Gamma}$ with $\mathcal{A} \models D(a)$, then $\mathcal{A}, \mathcal{T} \models A_0(a)$.*

Proof. We first show the following

Claim 1. For all $(C, S) \in \Gamma$ we have $\mathcal{T} \models \bigsqcap S \sqsubseteq C$.

Proof of Claim 1. Let $\Gamma_0, \dots, \Gamma_k = \Gamma$ be the sets of pairs obtained by repeatedly applying rules (r1) and (r2) to the initial set $\Gamma_0 = \{(A_0, \{A_0\})\}$. We show by induction on i that for all $(C, S) \in \Gamma_i$ we have $\mathcal{T} \models \bigsqcap S \sqsubseteq C$. The induction start is trivial. For the inductive step, first suppose that Γ_{i+1} is obtained from Γ_i by applying Rule (r1). Then there are $(C, S) \in \Gamma_i$ and $D \sqsubseteq A \in \mathcal{T}$ with $A \in S$ and such that Γ_{i+1} is Γ_i extended with the following tuples:

(a) $(C, S \setminus \{A\} \cup \mathsf{tlc}(D))$;
(b) for every $\exists r.G \in \mathsf{sub}(D)$, the pair $(G, \{G\})$.

The claim is trivially true for tuples added in (b). For the tuple in (a), the claim is a consequence of IH and the semantics.

Now assume that Γ_{i+1} is obtained from Γ_i by applying Rule (r2). Then there are $(C, S) \in \Gamma$ and $\mathcal{T} \models D \sqsubseteq \exists r.F \in \mathcal{T}$ such that Γ_{i+1} is Γ_i extended with the following tuples:

(a) $(C, (S \setminus \{\exists r.G \mid \mathcal{T} \models F \sqsubseteq G\}) \cup \mathsf{tlc}(D)$;
(b) for every $\exists r.G \in \mathsf{sub}(D)$, the pair $(G, \{G\})$.

Again, the claim is a consequence of the IH and semantics. This finishes the proof of the claim.

Using Claim 1 one can now prove the corresponding result for $\widehat{\Gamma}$ by induction over i for all $\widehat{\Gamma}_i$.

Claim 2. For all $(C, D) \in \widehat{\Gamma}$ we have $\mathcal{T} \models D \sqsubseteq C$.

Claim 2 directly implies Proposition 3: assume there is a $(A_0, D) \in \widehat{\Gamma}$ with $\mathcal{A} \models D(a)$. By Claim 2, $\mathcal{T} \models D \sqsubseteq A_0$. Thus $\mathcal{A}, \mathcal{T} \models A_0(a)$, as required. ❑

For the completeness part of Proposition 3 we require some preparation. First, for every $(C, D) \in \widehat{\Gamma}$ and variable $x \in \mathsf{var}(D)$, we denote by $\mu_{C,D}(x)$ the element of Γ used in the construction of D at variable x. Thus, for every $(C, S) \in \Gamma$ with $S \subseteq \mathsf{N_C}$ we set $\mu_{C,\bigsqcap S}(a_\varepsilon) = (C, S)$ and if $(C, D) \in \widehat{\Gamma}_{i+1}$ is constructed from some $(C, S) \in \Gamma$ and $(G, C_{r,G}) \in \widehat{\Gamma}_i$ for $\exists r.G \in S$ by putting $D = \bigsqcap(S \cap \mathsf{N_C}) \sqcap \bigsqcap_{\exists r.G \in S} \exists r.C_{r,G}$, then we let $\mu_{C,D}(x_\varepsilon) = (C, S)$ and $\mu_{C,D}(x) = \mu_{G,C_{r,G}}(x)$ for $x \in \mathsf{var}(C_{r,G})$.

We also need a suitable version of the chase procedure that reflects our rules (r1) and (r2) for constructing Γ. This chase introduces new ABox elements that are called *nulls*. It applies the following two rules:

(Ch1) If $\mathcal{A} \models C(a)$, a is not a null, $C \sqsubseteq A \in \mathcal{T}$, A a concept name, and $A(a) \notin \mathcal{A}$, then add $A(a)$ to $\mathcal{A}$;

(Ch2) If $\mathcal{A} \models C(a)$, a is not a null, $C \sqsubseteq \exists r.F \in \mathcal{T}$, and $\mathcal{A} \not\models \exists r.F(a)$, then add $r(a, b)$ to $\mathcal{A}$ with b a fresh null; further add the sub-ABox $\mathcal{A}_E$ (using only fresh nulls) rooted at b for every $\exists r.E \in \mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models F \sqsubseteq E$.

We set $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$ if there exists a sequence of ABoxes $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ such that each $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by a single application of rule (Ch1) or (Ch2) and $A_0(a) \in \mathcal{A}_k$. We then say that the sequence $\mathcal{A}_0, \ldots, \mathcal{A}_k$ *demonstrates* that $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$.

Lemma 4. *Let $\mathcal{T}$ be a TBox in rhs-nf and $\mathcal{A}$ an ABox with $a \in \mathsf{Ind}(\mathcal{A})$. Then $\mathcal{A}, \mathcal{T} \models A_0(a)$ iff $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$.*

We are now in the position to prove the completeness part of Proposition 3.

Lemma 5 (Completeness of $\widehat{\Gamma}$). *For all ABoxes $\mathcal{A}$ and $a \in \mathsf{Ind}(\mathcal{A})$, if $\mathcal{A}, \mathcal{T} \models A_0(a)$, then there is an $(A_0, D) \in \widehat{\Gamma}$ with $\mathcal{A} \models D(a)$.*

Proof. Assume that $\mathcal{A}, \mathcal{T} \models A_0(a)$. Then $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$ and consequently there is a sequence of ABoxes $\mathcal{A} = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k$ that demonstrates $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$. It thus suffices to prove by induction on k that

(∗) if $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ is a sequence that demonstrates $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$, then $\mathcal{A} \models D(a)$ for some $(A_0, D) \in \widehat{\Gamma}$.

The induction start is trivial: for $k = 0$, $A_0(a) \in \mathcal{A}_k$ implies $A_0(a) \in \mathcal{A}$, and clearly we have $(A_0, A_0) \in \widehat{\Gamma}$. For the induction step, assume that $\mathcal{A} = \mathcal{A}_0, \ldots, \mathcal{A}_k$ is a chase sequence that demonstrates $A_0(a) \in \mathsf{chase}_\mathcal{T}(\mathcal{A})$, with $k > 0$. Applying IH to the subsequence $\mathcal{A}_1, \ldots, \mathcal{A}_k$, we find an $(A_0, C) \in \widehat{\Gamma}$ with $\mathcal{A}_1 \models C(a)$.

Assume first that $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by rule (Ch1); i.e., by choosing a CI $D \sqsubseteq A \in \mathcal{T}$, A a concept name, and an individual $b \in \mathsf{ind}(\mathcal{A}_0)$ with $\mathcal{A}_0 \models D(b)$, and adding $A(b)$. Let h be a homomorphism from C to $\mathcal{A}_1$ with $h(x_\varepsilon) = a$ and let $x_1, \ldots, x_k$ be all elements of $\mathsf{var}(C)$ such that $A(x_i) \in C$ and $h(x_i) = b$. If this list of elements is empty, then $\mathcal{A}_0 \models C(a)$ and we are done. Otherwise, let the concept C' be obtained from C by replacing A with the concept D at every x_i, that is:

1. remove $A(x_i)$;
2. add $B(x_i)$ for each concept name $B \in \mathsf{tlc}(D)$;
3. add the subconcept $\exists r.F$ rooted at x_i for each $\exists r.F$ that is a tlc in D.

We show that $(A_0, C') \in \widehat{\Gamma}$ and $\mathcal{A}_0 \models C'(a)$, starting with the former. Fix some x_i and consider the pair $\mu_{A_0,C}(x_i) = (F_i, S_i) \in \Gamma$. By construction of $\widehat{\Gamma}$ and since $A(x_i) \in C$, we must have $A \in S_i$. Therefore, Rule (r1) from the construction of Γ is applicable to (F_i, S_i) and yields the pair

$$p_i = (F_i, S_i \setminus \{A\} \cup \mathsf{tlc}(D)) \in \Gamma.$$

When building up the pair $(A_0, C) \in \widehat{\Gamma}$ and dealing with node x_i, we can use the pair p_i in place of (F_i, S_i) and then proceed in a way such that the subtree rooted at x is D. From this, we obtain $(A_0, C') \in \widehat{\Gamma}$ as required.

Now we show $\mathcal{A}_0 \models C'(a)$. Since $\mathcal{A}_0 \models D(b)$, there is a homomorphism h' from D to $\mathcal{A}_0$ that maps x_ε to b. We obtain a homomorphism from C' to $\mathcal{A}_0$ that maps x_ε to a by starting with the homomorphism h and 'plugging in' h' at every node x_i to cover the subtrees generated by the subconcepts $\exists r.F$ of D added in Step 3 above.

Assume now that $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by an application of rule (Ch2); i.e., by choosing a CI $D \sqsubseteq \exists r.F \in \mathcal{T}$, and an individual $b \in \mathsf{ind}(\mathcal{A}_0)$ with $\mathcal{A}_0 \models D(b)$, and adding $r(b, c)$ and the sub-ABox $\mathcal{A}_E$ rooted at c for every $\exists r.E \in \mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models F \sqsubseteq E$. Let h be a homomorphism from C to $\mathcal{A}_1$ with $h(x_\varepsilon) = a$ and let $x_1, \dots, x_k$ be all elements of $\mathsf{var}(C)$ such that $h(x_i) = b$ and there is an x_i' with $r(x_i, x_i') \in C$ and $h(x_i') = c$. If this list of elements is empty, then $\mathcal{A}_0 \models C(a)$ and we are done. Otherwise, let the concept C' be obtained from C by replacing all $\exists r.G$ with $\mathcal{T} \models F \sqsubseteq G$ with D at every x_i, that is:

1. if $r(x_i, x_i') \in C$ and $\mathcal{T} \models F \sqsubseteq C|_{x_i'}$, then remove from C the edge $r(x_i, x_i')$ and the subtree rooted at x_i';
2. add $B(x_i)$ for each concept name $B \in \mathsf{tlc}(D)$;
3. add the subconcept $\exists s.E$ rooted at x_i for each $\exists s.E$ that is a tlc in D.

We show that there is a concept C'' such that (i) there is a homomorphism from C'' to C' and $(A_0, C'') \in \widehat{\Gamma}$ and (ii) $\mathcal{A}_0 \models C'(a)$, starting with (i). Fix some x_i and let $x_1^i, \dots, x_{k(i)}^i$ be all individual variables such that $r(x_i, x_j^i) \in C$ and $h(x_j^i) = c$. By construction of $\mathcal{A}_1$, and since $h(x_j^i) = c$, for $1 \leq j \leq k(i)$, we must have $\mathcal{T} \models F \sqsubseteq C|_{x_j^i}$. Now consider the pair $\mu_{A_0,C}(x_i) = (F_i, S_i) \in \Gamma$. Let $1 \leq j \leq k(i)$. By construction of $\widehat{\Gamma}$ and since $(A_0, C) \in \Gamma$ and $r(x_i, x_j^i) \in C$, there must be an $\exists r.G_j \in S_i$ such that $(G_j, C|_{x_j^i}) \in \widehat{\Gamma}$. By Claim 1 from the proof of Lemma 3, this yields $\mathcal{T} \models C|_{x_j^i} \sqsubseteq G_j$. Together with $\mathcal{T} \models F \sqsubseteq C|_{x_j^i}$, we obtain $\mathcal{T} \models F \sqsubseteq G_j$. Thus Rule (r2) from the construction of Γ is applicable to (F_i, S_i) and yields the pair

$$p_i = (F_i, (S_i \setminus \{\exists r.G_j \mid 1 \leq j \leq k(i)\}) \cup \mathsf{tlc}(D).$$

When constructing the pair $(A_0, C) \in \widehat{\Gamma}$ and dealing with node x_i, we can use the pair p_i in place of (F_i, S_i) and then proceed in a way such that the subtree rooted at x_i is $D \sqcap \bigsqcap S_i \setminus (\mathsf{tlc}(D) \cup \{\exists r.G_j \mid 1 \leq j \leq k(i)\})$, in this way obtaining a pair $(A_0, C'') \in \widehat{\Gamma}$. It can be verified that there is a homomorphism of C'' (which amounts to checking that there is a homomorphism from $C''|_{x_i}$ to $C'|_{x_i}$; in fact, we have $\mathsf{tlc}(C'') \subseteq \mathsf{tlc}(C')$), and thus we are done with proving (i).

For (ii), we have to show that $\mathcal{A}_0 \models C'(a)$. Since $\mathcal{A}_0 \models D(b)$, there is a homomorphism h' from D to $\mathcal{A}_0$ that maps x_ε to b. We obtain a homomorphism of from C' to $\mathcal{A}_0$ that maps x_ε to a by starting with the homomorphism h and 'plugging in' h' at every node x_i to cover the subtrees generated by the subconcepts $\exists s.E$ of D added in Step 3 above. ❑

Proposition 4. If Ω contains a root cycle, then A_0 is not FO-rewritable under $\mathcal{T}$.

Proof. Assume that Ω contains a root cycle.. We show that there exists a blocked concept C with $\mathcal{T} \models C \sqsubseteq A_0$ as well as $\mathcal{T} \not\models (C \setminus C|_{x_3}) \sqsubseteq A_0$, where x_3 is as in the definition of blocked concepts. Once we have constructed such a concept C, non-FO-rewritability follows as in the proof of Proposition 2.

We will construct the desired concept C as the limit of a finite sequence of concepts $C_0, \ldots, C_m$. Along with this concept sequence, we construct a sequence of mappings $\mu_0, \ldots, \mu_m$ where μ_i associates with each individual variable in $\mathsf{var}(C_i)$ a tuple from Ω. To start the construction, choose a root tuple $t_\varepsilon \in \Omega$ and a looping tuple $t_{\mathsf{loop}} \in \Omega$ that is reachable from t_ε along $\rightsquigarrow_\Omega$. Set $C_0 = \{A \in \mathsf{N_C} \mid A \in S_{t_\varepsilon}\}$, and $\mu_0(x_\varepsilon) = t_\varepsilon$.

Now assume that C_ℓ is already defined. To construct $C_{\ell+1}$, choose an $x \in \mathsf{var}(C_\ell)$ with $\mu_\ell(x) = (C_x, S_x, \mathsf{con}_x, s_x, \mathsf{xcon}_x)$ such that the set of existential restrictions in S_x is non-empty but x does not yet have any successors in C_ℓ. Since the set of existential restrictions in S_x is non-empty the tuple $\mu_\ell(x)$ is not a leaf tuple and by the construction of Ω this implies that there are tuples $t_0, \ldots, t_n \in \Omega$ such that Rule (r3) *can be applied to* $t_0, \ldots, t_n$ *and selected successor* $\ell \leq n$ *to generate* $\mu_\ell(x)$; i.e., for $t = \mu_\ell(x)$ such that $\exists r_0.D_0, \ldots, \exists r_n.D_n$ are the existential restrictions in S_t and $s_t = \exists r_\ell.D_\ell$ we have $C_{t_i} = D_i$ for $0 \leq i \leq n$ and:

- there is a node pair $(C_t, S) \in \Gamma$ with $S_t \subseteq S$ and $S \cap \mathsf{N_C} = S_t \cap \mathsf{N_C}$;;
- $\mathsf{con}_t = \mathsf{con}_\mathcal{T}(M)$, where $M = \bigcup(S_t \cap \mathsf{N_C}) \cup \{\exists r_i.G \mid i \leq n \text{ and } G \in \mathsf{con}_{t_i}\}$;
- $\mathsf{xcon}_t = \mathsf{con}_\mathcal{T}(M')$, where M' is

$$\bigcup(S_t \cap \mathsf{N_C}) \cup \{\exists r_\ell.G \mid G \in \mathsf{xcon}_{t_\ell}\} \cup \{\exists r_i.G \mid \ell \neq i \leq n \text{ and } G \in \mathsf{con}_{t_i}\}.$$

To construct $C_{\ell+1}$, add $r_i(x, y_i)$ to C_ℓ, for fresh individual variables y_i and all $i \leq n$. Further add the assertion $A(y_i)$ for each $A \in S_{t_i} \cap \mathsf{N_C}$. The resulting concept is $C_{\ell+1}$. Finally, $\mu_{\ell+1}$ is μ_ℓ extended by setting $\mu_{\ell+1}(y_i) = t_i$ for all $i \leq n$.

Unless guided in an appropriate way, the above construction need not terminate and will not result in a concept that has the desired properties. Let

$\Omega_0, \Omega_1, \ldots \Omega_k$ with $\Omega_k = \Omega$ be the sequence of sets generated by repeated application of Rule (r3). For each $t \in \Omega$, let $\mathsf{rank}(t)$ denote the smallest i such that $t \in \Omega_i$. We guide, in the above construction, the selection of the tuples from Ω in the following way.

Step 1. We construct $C_0, C_1, \ldots, C_{m_1}$ such that $\mu_{m_1}(x_1) = t_{\mathsf{loop}}$ for some leaf variable x_1 of C_{m_1}. We know that t_{loop} is reachable from t_ε along $\rightsquigarrow_\Omega$. We start with t_ε as before but at each step ℓ, we choose $t_0, \ldots, t_n$ such that Rule (r3) can be applied to $t_0, \ldots, t_n$ and selected successor $j \leq n$ to generate $\mu_\ell(x)$ so that t_j brings us closer to t_{loop}, that is, the shortest $\rightsquigarrow_\Omega$-path from t_j to t_{loop} is shorter than the shortest such path from $\mu_\ell(x)$ to t_{loop}. Let y_j be the individual that we introduced as a successor of x with $\mu_{\ell+1}(y_j) = t_j$. In step $\ell + 1$, we continue from y_j.

Step 2. Step 1 guarantees that we have $C_0, C_1, \ldots, C_{m_1}$ with $\mu_{m_1}(x_1) = t_{\mathsf{loop}}$ for some leaf variable x_1 of C_{m_1}. In Step 2 we extend the sequence $C_0, C_1, \ldots, C_{m_1}$ by $C_{m_1+1}, \ldots, C_{m_2}$ in such a way that $\mu_{m_2}(x_2) = t'$ for some leaf variable x_2 of C_{m_2} such that the con and xcon components of t_{loop} and t' coincide. Since t_{loop} is looping, there is a tuple $t' \in \Omega$ such that t' is reachable from t_{loop} on a $\rightsquigarrow_\Omega$-path and the con and xcon components of t_{loop} agree with those of t'. To construct $C_{m_1+1}, \ldots, C_{m_2}$, we start with x_1 and follow the same strategy as in Step 1, but this time to reach t' instead of t_{loop}.

Step 3. Let $C_0, C_1, \ldots, C_{m_2}$ be the sequence constructed in Step 2. To finish the construction of C we expand this sequence as follows. Assume $n \geq m_2$, C_n has been constructed, and $x \in \mathsf{var}(C_n)$ is such that x does not have any successors in C_n but there is some $\exists r.D \in S_x$. Then we choose $t_0, \ldots, t_n$ such that Rule (r3) can be applied to $t_0, \ldots, t_n$ and selected successor $j \leq n$ to generate $\mu_\ell(x)$ so that $t_0, \ldots, t_n$ brings us closer to a leaf tuple, i.e., $\mathsf{rank}(t_i) < \mathsf{rank}(\mu_\ell(x))$ for all $i \in \{0, \ldots, n\}$. Note that, by construction of Ω, this is always possible.

It should be clear that this way of guiding tuple selection guarantees termination. Call the concept and mapping obtained in the limit C and μ, respectively. For each $x \in C$, p_x is defined as the (unique) path $x_0, \ldots, x_k$ such that $x_0 = x$, x_k is a leaf of C, and for all $i \in \{0, \ldots, k-1\}$, if $t_0, \ldots, t_n$ are the tuples that we picked in the inductive construction above while adding successors of x_i using Rule (r3) with selected successor $j \leq n$, then we have $\mu(x_{i+1}) = t_j$.

Claim. For all $x \in \mathsf{var}(C)$ we have

1. $\mathsf{con}_{\mu(x)} = \mathsf{con}^C_{\mathcal{T}}(x)$;
2. if $y \neq x$ is the last element of p_x, then $\mathsf{xcon}_{\mu(x)} = \mathsf{con}^{C \setminus C|_y}_{\mathcal{T}}(x)$.

Proof of claim. The proof is by induction on the length of the path p_x. As the base case, p_x is of length one and thus, $\mu(x)$ is a leaf tuple. By definition, this implies that S_x consists of concept names. Let $S_{\mu(x)} = \{A_1, \ldots, A_k\}$. By our construction, $C|_x = A_1 \sqcap \cdots \sqcap A_k$. Let $D \in \mathsf{sub}(\mathcal{T})$. By definition, $D \in \mathsf{con}_{\mu(x)}$ iff $\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_k \sqsubseteq D$, which is Point 1 in the claim. Since $\mu(x)$ is a leaf, we have $y = x$ for the last element y of p_x. Hence Point 2 in the claim holds trivially and the base case is proved.

For the inductive step, suppose p_x is of length $i > 1$. Let $t_0, \dots, t_n$ be the tuples that we picked in the inductive construction of C while adding successors $y_0, \dots, y_n$ of x using Rule (r3) with selected successor $\ell \leq n$. Assume the existential restrictions in $S_{\mu(x)}$ are $\exists r_0.D_0, \dots, \exists r_n.D_n$ and we have

- $r_k(x, y_k) \in C$ and $C_{\mu(y_k)} = D_k$ for $0 \leq k \leq n$.
- $\mathsf{con}_{\mu(x)} = \mathsf{con}_{\mathcal{T}}(M)$, where

$$M = \bigsqcap (S_{\mu(x)} \cap \mathsf{N_C}) \cup \{\exists r_k.G \mid k \leq n \text{ and } G \in \mathsf{con}_{\mu(y_k)}\}$$

- $\mathsf{xcon}_{\mu(x)} = \mathsf{con}_{\mathcal{T}}(M')$, where

$$M' = \bigsqcap (S_{\mu(x)} \cap \mathsf{N_C}) \cup \{\exists r_\ell.G \mid G \in \mathsf{xcon}_{\mu(y_\ell)}\} \cup \{\exists r_k.G \mid \ell \neq k \leq n \text{ and } G \in \mathsf{con}_{\mu(y_k)}\}.$$

Observe that $C|_x = \bigsqcap (S_{\mu(x)} \cap \mathsf{N_C}) \sqcap \exists r_0.C|_{y_0} \sqcap \cdots \exists r_n.C|_{y_n}$. The IH implies

- $\mathsf{con}_{\mu(y_k)} = \mathsf{con}^{C}_{\mathcal{T}}(y_k)$ for all $k \leq n$;
- $\mathsf{xcon}_{\mu(y_\ell)} = \mathsf{con}^{C \setminus C|_y}_{\mathcal{T}}(y_\ell)$ for the last element y of p_x.

Hence we obtain with Lemma 2 and the IH that $\mathsf{con}_{\mu(x)} = \mathsf{con}^{C}_{\mathcal{T}}(x)$ and $\mathsf{xcon}_{\mu(x)} = \mathsf{con}^{C \setminus C|_y}_{\mathcal{T}}(x)$, as required. ⊣

Recall that there is a path $x_0, \dots, x_k \in \mathsf{var}(C)$ such that x_0 is the root of C, x_k is a leaf of C, for all $i < k$, x_{i+1} represents the selected successor of $\mu(x_i)$, and there are nodes x_p and x_q on the path such that $p < q$, $\mu(x_p) = t_{\mathsf{loop}}$, and $\mu(x_q) = t'$. By the claim we have

- $\mathsf{con}^{C}_{\mathcal{T}}(x_p) = \mathsf{con}_{\mu(x_p)} = \mathsf{con}_{\mu(x_q)} = \mathsf{con}^{C}_{\mathcal{T}}(x_q)$;
- $\mathsf{con}^{C \setminus C|_{x_k}}_{\mathcal{T}}(x_p) = \mathsf{xcon}_{\mu(x_p)} = \mathsf{xcon}_{\mu(x_q)} = \mathsf{con}^{C \setminus C|_{x_k}}_{\mathcal{T}}(x_q)$.

It follows that C is blocked and $\mathcal{T} \models C \sqsubseteq A_0$ but $\mathcal{T} \not\models C \setminus C|_{x_k} \sqsubseteq A_0$, as required. ❑

We fix some notation for the semantics of datalog programs. Assume Π is a monadic datalog program with goal predicate $G(x)$. For an ABox $\mathcal{A}$ we define a sequence of ABoxes $\Pi^0(\mathcal{A}), \Pi^1(\mathcal{A}), \dots$ by setting:

- $\Pi^0(\mathcal{A}) = \mathcal{A}$;
- $\Pi^{n+1}(\mathcal{A})$ is defined by adding to $\Pi^n(\mathcal{A})$ the set of all $P(a)$ with $a \in \mathsf{ind}(\mathcal{A})$ and P an IDB of Π such that there exists a rule $P(x) \leftarrow \varphi$ in Π and a variable assigment π with $\pi(x) = a$ and $\Pi^n(\mathcal{A}) \models_\pi \varphi$.

Set $\Pi(\mathcal{A}) = \bigcup_{n \geq 0} \Pi^n(\mathcal{A})$. We use $\mathsf{con}^{\mathcal{A}}_{\mathcal{T}}(a)$ to denote $\{D \in \mathsf{sub}(\mathcal{T}) \mid \mathcal{A}, \mathcal{T} \models D(a)\}$.

Theorem 1.

1. The program $\Pi_{\mathcal{T}, A_0}$ is a rewriting of A_0 under $\mathcal{T}$.

2. If Ω contains no root cycle, then $\Pi_{\mathcal{T},A_0}$ is non-recursive.

Proof. For Point 1, first assume $P_{A_0,\mathsf{con}_0}(a_0) \in \Pi_{\mathcal{T},A_0}(\mathcal{A})$ for a goal predicate P_{A_0,con_0} of $\Pi_{\mathcal{T},A_0}$. We have to show that $\mathcal{A},\mathcal{T} \models A_0(a_0)$. The following claim can be proved by induction using the construction of Ω:

Claim. For all $n \geq 0$, all $a \in \mathsf{ind}(\mathcal{A})$, and all IDBs $P_{C,\mathsf{con}}$ of $\Pi_{\mathcal{T},A_0}$ with $P_{C,\mathsf{con}}(a) \in \Pi^n(\mathcal{A})$ we have $\mathsf{con} \subseteq \mathsf{con}^{\mathcal{A}}_{\mathcal{T}}(a)$.

Now, since $P_{A_0,\mathsf{con}_0}(a_0) \in \Pi_{\mathcal{T},A_0}(\mathcal{A})$ and $A_0 \in \mathsf{con}_0$ (by the definition of goal predicates in $\Pi_{\mathcal{T},A_0}$) we obtain $A_0 \in \mathsf{con}^{\mathcal{A}}_{\mathcal{T}}(a_0)$, as required.

Conversely, assume that $\mathcal{A},\mathcal{T} \models A_0(a_0)$. We have to show that $G(a_0) \in \Pi_{\mathcal{T},A_0}(\mathcal{A})$ for a goal predicate G of $\Pi_{\mathcal{T},A_0}$. By Proposition 3, there is an $(A_0, C') \in \widehat{\Gamma}$ with $\mathcal{A} \models C'(a_0)$. Let $C \preceq^* C'$ be minimal with $\mathcal{T} \models C \sqsubseteq A_0$. Since C homomorphically maps to $\mathcal{A}$, it suffices to show that $G(x_\varepsilon) \in \Pi_{\mathcal{T},A_0}(\mathcal{A}_C)$ with x_ε the root of $\mathcal{A}_C$. Note that C can be obtained from C' by removing subtrees. Thus, each node in C is in the domain of the function $\mu_{A_0,C'}$. For brevity, when $x \in \mathsf{var}(C)$ and $\mu_{A_0,C'}(x) = (D, S)$, we use C_x to denote D and S_x to denote S.

We aim to show that for each $x \in \mathsf{var}(C)$, the following rule R_x is in $\Pi_{\mathcal{T},A_0}$:

$$P_{C_x,\mathsf{con}^C_{\mathcal{T}}(x)}(v) \leftarrow \bigwedge_{A \in S_x \cap \mathsf{N_C}} A(v) \wedge \bigwedge_{r(x,y) \in C} (r(v,w) \wedge P_{C_y,\mathsf{con}^C_{\mathcal{T}}(x)}(w)).$$

Note that, by definition of $\mu_{A_0,C'}$, $A \in S_x \cap \mathsf{N_C}$ implies $A(x) \in \mathcal{A}_C$. Thus if we have shown that the above rules are all in $\Pi_{\mathcal{T},A_0}$, it is easy to prove by induction on the co-depth of x that for all $x \in \mathsf{var}(C)$, we have $P_{C_x,\mathsf{con}^C_{\mathcal{T}}(x)}(x) \in \Pi_{\mathcal{T},A_0}(\mathcal{A}_C)$. From $\mathcal{A}_C,\mathcal{T} \models A_0(x_\varepsilon)$, we obtain $A_0 \in \mathsf{con}^C_{\mathcal{T}}(x_\varepsilon)$; moreover, the definition of $\mu_{A_0,C'}$ yields $C_{x_\varepsilon} = A_0$. Consequently, $P_{C_{x_\varepsilon},\mathsf{con}^C_{\mathcal{T}}(x_\varepsilon)}(x_\varepsilon)$ is a goal predicate of $\Pi_{\mathcal{T},A_0}$ and we are done.

We now exhibit a way to decorate the individuals in $\mathcal{A}_C$ with tuples from Ω. Fix any choice function f that assigns to each non-leaf $x \in \mathsf{ind}(\mathcal{A}_C)$ an atom $r(x,y) \in \mathcal{A}_C$. Then associate with each node $x \in \mathsf{ind}(\mathcal{A}_C)$ a node tuple $\zeta_f(x)$ in a bottom-up way as follows:

(a) for each leaf $x \in \mathsf{ind}(\mathcal{A}_C)$, set

$$\zeta_f(x) = (C_x, S_x \cap \mathsf{N_C}, \mathsf{con}_{\mathcal{T}}(S_x), -, -).$$

(b) for each non-leaf node $x \in \mathsf{ind}(\mathcal{A}_C)$, set

$$\zeta_f(x) = (C_x, S, \mathsf{con}_{\mathcal{T}}(M), s, \mathsf{con}_{\mathcal{T}}(M'))$$

where

- $S = (S_x \cap \mathsf{N_C}) \cup \{\exists r.E \in S \mid r(x,y) \in \mathcal{A}_C,\ C_y = E\}$;
- $M = (S_x \cap \mathsf{N_C}) \cup \{\exists r.G \mid r(x,y) \in \mathcal{A}_C,\ G \in \mathsf{con}_{\zeta_f(y)}\}$;
- for $f(x) = \widehat{r}(x,\widehat{y})$,
  - $s = \exists \widehat{r}.C_{\widehat{y}}$ and

- $M' = (S_x \cap \mathsf{N_C}) \cup \{\exists \widehat{r}.G \mid G \in \mathsf{xcon}_{\zeta_f(\widehat{y})}\} \cup$
  $\{\exists r.G \mid G \in \mathsf{con}_{\zeta_f(y)}, r(x,y) \in \mathcal{A}_C \text{ and } y \neq \widehat{y}\}$.

Observe that for each non-leaf x, there exists y with $r(x,y) \in \mathcal{A}_C$ and $\zeta_f(x) \rightsquigarrow_\Omega \zeta_f(y)$. Thus one can show by induction on the co-depth of x in $\mathcal{A}_C$ that $\zeta_f(x)$ was in Ω before we have dropped tuples that were not reachable from a root tuple. Let $x_0, \ldots, x_m$ be the path through $\mathcal{A}_C$ from the root to a leaf that is obtained by following the successors selected by f. It follows from Lemma 2 that

(i) $\mathsf{con}_{\mathcal{T}}^{\mathcal{A}_C}(x) = \mathsf{con}_{\zeta_f(x)}$, for all $x \in \mathsf{ind}(\mathcal{A}_C)$;
(ii) $\mathsf{con}_{\mathcal{T}}^{\mathcal{A}_{C \setminus C|_{x_m}}}(x_i) = \mathsf{xcon}_{\zeta_f(x_i)}$, for all $i < m$.

Thus, since $\mathcal{A}_C, \mathcal{T} \models A_0(x_0)$, we have $A_0 \in \mathsf{con}_{\zeta_f(x_0)}$. Since C is minimal with $\mathcal{T} \models C \sqsubseteq A_0$, we have $\mathcal{A}_{C \setminus C|_{x_m}}, \mathcal{T} \not\models A_0(x_0)$ and so $A_0 \notin \mathsf{xcon}_{\zeta_f(x_0)}$. Consequently, $\zeta_f(x_0)$ is a root tuple. Since the above holds for *any* choice of f, we find for each $x \in \mathsf{ind}(\mathcal{A}_C)$ a choice function f such that $\zeta_f(x)$ is in Ω also *after* dropping all node tuples that are not reachable from a root tuple: simply choose f so that x is reachable from the root along successors selected by f. Also note that, by Point (i) above and the construction of the decorations ζ_f, the first three components of the tuple $\zeta_f(x)$ are independent of f.

It is now also easy to show that for all $x \in \mathsf{ind}(\mathcal{A}_C)$, the rule R_x is in $\Pi_{\mathcal{T},A_0}$. In fact, as we have seen above, there is a choice function f with $\zeta_f(x) \in \Omega$. Consider the rule R corresponding in $\Pi_{\mathcal{T},A_0}$ to $t = \zeta_f(x)$. It is easily verified that the conjunctions in R_x and R range over the same sets. It thus remains to be verified that for every $r(x,y) \in \mathcal{A}_C$, there is a $t' \in \Omega$ such that $t \rightsquigarrow_{\exists r.C_y} t'$ and $\mathsf{con}_{t'} = \mathsf{con}_{\mathcal{T}}^{\mathcal{A}_C}(y)$. In other words, we have to show that there are $\widehat{t} = (C_t, S_t, \mathsf{con}_t, \exists r.C_y, \mathsf{xcon}_{\widehat{t}}) \in \Omega$ and $t' \in \Omega$ such that $\mathsf{con}_{t'} = \mathsf{con}_{\mathcal{T}}^{\mathcal{A}_C}(y)$ and $\widehat{t} \rightsquigarrow_\Omega t'$. To identify these tuples, select f' so that y is reachable from the root along successors selected by f'. Then $\zeta_{f'}(z)$ is in Ω for all nodes z on the path between x_0 and y, including for $z = x$. Now set $\widehat{t} = \zeta_{f'}(x)$ and $t' = \zeta_{f'}(y)$. It can be verified that $\widehat{t}$ and t' are as required.

It remains to establish Point 2. Assume that Ω contains no root cycle, but, to the contrary of what is to be shown, there are rules $P_{C_0,\mathsf{con}_0}(x) \leftarrow \varphi_0(x), \ldots, P_{C_n,\mathsf{con}_n}(x) \leftarrow \varphi_n(x)$ such that (i) $P_{C_0,\mathsf{con}_0} = P_{C,\mathsf{con}}$ and (ii) P_{C_i,con_i} occurs in φ_{i+1} for all $i \leq n$ and with $\varphi_{n+1} = \varphi_0$. We show that this gives rise to a path through Ω that starts at a root tuple and has length exceeding $2^{2|\mathcal{T}|}$. Since any such path must contain a root cycle we have derived a contradiction to the fact that no such cycle exists.

We construct the path by traveling backwards along "$\rightsquigarrow_\Omega$". With each tuple t on the path, we associate a rule $P_{C_i,\mathsf{con}_i}(x) \leftarrow \varphi_i(x)$ such that $C_t = C_i$ and $\mathsf{con}_t = \mathsf{con}_i$. Start the path with some tuple $t_0 \in \Omega$ that gives rise to the rule $P_{C_0,\mathsf{con}_0}(x) \leftarrow \varphi_0(x)$ (note that different tuples might give rise to the same rule). Clearly, we have $C_{t_0} = C_0$ and $\mathsf{con}_{t_0} = \mathsf{con}_0$ as required. Assume that an initial piece $t_k \rightsquigarrow_\Omega \cdots \rightsquigarrow_\Omega t_0$ of the path through Ω has already been constructed. To extend it, let $P_{C_i,\mathsf{con}_i}(x) \leftarrow \varphi_i(x)$ be the rule associated with t_k. Then we have $C_{t_k} = C_i$ and $\mathsf{con}_{t_k} = \mathsf{con}_i$. Let t be some tuple that gives rise

to $P_{C_{i+1},\mathsf{con}_{i+1}}(x) \leftarrow \varphi_{i+1}(x)$. Then $C_t = C_{i+1}$ and $\mathsf{con}_t = \mathsf{con}_{i+1}$. Since P_{C_i,con_i} occurs in φ_{i+1}, there must be an $\exists r.D \in S_t$ and a tuple $t' \in \Omega$ with $t \rightsquigarrow_{\exists r.D} t'$ and such that $C_{t'} = D = C_i$ and $\mathsf{con}_{t'} = \mathsf{con}_i$. By definition of "$\rightsquigarrow_{\exists r.D}$", there is a tuple $\widehat{t} = (C_t, S_t, \mathsf{con}_t, \exists r.D, \mathsf{xcon}_{\widehat{t}})$ such that $\widehat{t} \rightsquigarrow_\Omega t'$. By definition of "$\rightsquigarrow_\Omega$" and since $C_{t'} = C_{t_k} = C_i$ and $\mathsf{con}_{t'} = \mathsf{con}_{t_k} = \mathsf{con}_i$, we find a tuple $\widehat{t'}$ with $C_{\widehat{t'}} = C_{\widehat{t}}$, $\mathsf{con}_{\widehat{t'}} = \mathsf{con}_{\widehat{t}}$, and $\widehat{t'} \rightsquigarrow_\Omega t_k$ (note that $\widehat{t'}$ need not be the same as $\widehat{t}$ since the xcon-component might change). We use $\widehat{t'}$ as the next tuple t_{k+1} on our path through Ω. Note that $C_{\widehat{t'}} = C_{i+1}$ and $\mathsf{con}_{\widehat{t'}} = \mathsf{con}_{i+1}$ as required. Proceeding in this way, we can continue to build up a backwards path through Ω until its length exceeds $2^{2|\mathcal{T}|}$. It remains to further extend the path (backwards) towards a root tuple. This is trivially possible since we have dropped from Ω all tuples that are not reachable from a root tuple. ❑

Theorem 2. There is a family of TBoxes $\mathcal{T}_1, \mathcal{T}_2, \ldots$ such that for all $n \geq 1$, $\mathcal{T}_n$ is of size $\mathcal{O}(n^2)$, the concept name A_0 is FO-rewritable under $\mathcal{T}_n$, and the smallest non-recursive monadic datalog rewriting has size at least 2^n.

Proof. We first observe that one can assume w.l.o.g. that the monadic datalog programs are connected. We call a rule $P(x) \leftarrow \varphi$ *connected* if the graph whose vertices are x and the remaining variables in φ and whose edges are $\{x_1, x_2\}$ for $r(x_1, x_2) \in \varphi$ is connected. A monadic datalog program is *connected* if all its rules are connected.

Claim 1. If Π is a non-recursive monadic datalog rewriting of A_0 relative to $\mathcal{T}$, then there exists a connected non-recursive monadic datalog rewriting Π' of A_0 relative to $\mathcal{T}$ whose size does not exceed the size of Π.

To prove Claim 1 assume Π is given. Define Π' by replacing each rule $P(x) \leftarrow \varphi$ with the rule $P(x) \leftarrow \varphi'$, where φ' is obtained from φ by removing all atoms $A(y)$ and $r(x_1, x_2)$ whose variables are not connected to x in φ (thus, if x does not occur in φ then $P(x) \leftarrow \varphi$ is replaced by $P(x) \leftarrow$. We show that Π' is a datalog rewriting of A_0 relative to $\mathcal{T}$. Clearly $\Pi(\mathcal{A}) \subseteq \Pi'(\mathcal{A})$ for any ABox $\mathcal{A}$. Thus, it remains to show that if $G(a) \in \Pi'(\mathcal{A})$ for an ABox $\mathcal{A}$, $a \in \mathsf{ind}(\mathcal{A})$, and goal predicate G, then $G'(a) \in \Pi(\mathcal{A})$ for some goal predicate G'. Assume this is not the case. Take $\mathcal{A}$, $a \in \mathsf{ind}(\mathcal{A})$, and goal predicate G such that $G(a) \in \Pi'(\mathcal{A})$ but $G'(a) \notin \Pi(\mathcal{A})$ for any goal predicate G'. Define an ABox $\mathcal{A}'$ by adding to $\mathcal{A}$ the assertions $A(c)$ and $r(c, c)$ for every concept name A and role name r in Π and a fresh individual name c. Clearly $G(a) \in \Pi(\mathcal{A}')$ since the atoms removed from Π can all be satisfied in c. Hence $\mathcal{A}', \mathcal{T} \models A_0(a)$ since Π is a rewriting of A_0 relative to $\mathcal{T}$. But then $\mathcal{A}, \mathcal{T} \models A_0(a)$ since $\mathcal{T}$ is an $\mathcal{EL}$-TBox and c is disconnected from the individuals in $\mathcal{A}$. We have derived a contradiction to the assumption that Π is a rewriting of A_0 relative to $\mathcal{T}$. This finishes the proof of Claim 1.

We now provide a reformulation of Theorem 2. A *tree-UCQ-rewriting of* A_0 *relative to* $\mathcal{T}$ is a disjunction $\bigvee M$, where M is a finite set of $\mathcal{EL}$-concepts.

Claim 2. Theorem 2 follows if there is a family of TBoxes $\mathcal{T}_1, \mathcal{T}_2, \ldots$ such that for all $n \geq 1$, $\mathcal{T}_n$ is of size $\mathcal{O}(n^2)$, the concept name A_0 is tree-UCQ-rewritable

under $\mathcal{T}_n$, but any tree-UCQ-rewriting of A_0 under $\mathcal{T}_n$ contains a concept C of depth at least 2^n.

To prove Claim 2, let $\mathcal{T}_n$ be one of the TBoxes from the claim. Let Π be a non-recursive monadic datalog rewriting of A_0 w.r.t. $\mathcal{T}$. We have to show that Π has size at least 2^n. By Claim 1, we may assume that Π is connected. Let $\bigvee M$ be a tree-UCQ-rewriting of A_0 under $\mathcal{T}_n$. We may assume that each $C \in M$ is minimal (i.e., if $C' \preceq C$, then $\mathcal{T} \not\models C' \sqsubseteq A_0$). Since $\mathcal{T}_n$ is a TBox from the claim, we find a $C \in M$ whose depth is at least 2^n. We now have $G(x_\varepsilon) \in \Pi(\mathcal{A}_C)$ for the root x_ε of the ABox $\mathcal{A}_C$ corresponding to C. Using that assumptions that Π is connected and non-recursive and that C is minimal with $\mathcal{T} \models C \sqsubseteq A_0$ it follows immediately that there is a sequence of distinct rules $P_0(x) \leftarrow \varphi_0, \dots, P_m(x) \leftarrow \varphi_m$ in Π with P_{i+1} in φ_i for $i < m$ and $\sum_{i=1}^{m} |\varphi_i| \geq 2^n$, as required. This finishes the proof of Claim 2.

It thus remains to identify TBoxes $\mathcal{T}_1, \mathcal{T}_2, \dots$ with the properties of Claim 2. As an abbreviation, define $C_0 := B_0$ and $C_i := B_i \sqcap \exists r.C_{i-1}$. Let $n \geq 1$. Then $\mathcal{T}_n$ consists of the following CIs, for all $i < n$:

$$
\begin{aligned}
\exists r.(A_0 \sqcap B_i) &\sqsubseteq A_0 \\
\exists r.(A_0 \sqcap \overline{B}_i) &\sqsubseteq A_0 \\
B_i \sqcap \exists r.B_j &\sqsubseteq A_0 \quad \text{for } 1 \leq j < n \text{ with } i \neq j+1 \\
\overline{B}_i \sqcap \exists r.B_j &\sqsubseteq A_0 \quad \text{for } 1 \leq j < n \text{ with } i \neq j+1 \\
B_i \sqcap \exists r.\overline{B}_j &\sqsubseteq A_0 \quad \text{for } 1 \leq j < n \text{ with } i \neq j+1 \\
\overline{B}_i \sqcap \exists r.\overline{B}_j &\sqsubseteq A_0 \quad \text{for } 1 \leq j < n \text{ with } i \neq j+1 \\
B_i \sqcap \exists r^{n+1}.(B_i \sqcap \exists r.C_{i-1}) &\sqsubseteq A_0 \\
\overline{B}_i \sqcap \exists r^{n+1}.(\overline{B}_i \sqcap \exists r.C_{i-1}) &\sqsubseteq A_0 \\
B_i \sqcap \exists r^{n+1}.(\overline{B}_i \sqcap \exists r^j.\overline{B}_{i-j}) &\sqsubseteq A_0 \quad \text{for } 1 \leq j < i \\
\overline{B}_i \sqcap \exists r^{n+1}.(B_i \sqcap \exists r^j.\overline{B}_{i-j}) &\sqsubseteq A_0 \quad \text{for } 1 \leq j < i \\
C_{n-1} &\sqsubseteq A_0.
\end{aligned}
$$

Note that the concept names $B_0, \dots, B_{n-1}$ and $\overline{B}_0, \dots, \overline{B}_{n-1}$ are trivially FO-rewritable because they do not occur on the right-hand side of any CI. Because of the first two CIs, the concept name A_0 propagates 'backwards' along r-chains in an ABox, which in principle gives rise to non-FO-rewritability of A_0. FO-rewritability is regained, though, by enforcing the existence of a binary counter in the ABox via the concept names $B_0, \dots, B_{n-1}$ and $\overline{B}_0, \dots, \overline{B}_{n-1}$, which represent positive and negative bits, respectively. The counter is 'spaced out' in the sense that every individual on a backwards r-chain stores only a single bit of the counter. Concept inclusions 3-6 ensure that the bits appear in the right order and CIs 7-10 make sure that the counter is incremented properly, by otherwise entailing A_0, thus disrupting the unbounded propagation. The last concept inclusion entails A_0 if the counter value has reached maximum, thus stopping unbounded propagation after at most $2^n \cdot n$ steps.

A concrete tree-UCQ-rewriting for A_0 under $\mathcal{T}$ is as follows. For any concept C and $i \geq 0$, define a set of concepts $S[C]$ as follows: $S_0[C] = \{C\}$ and $S_{i+1}[C] := \{\exists r.(B_i \sqcap D), \exists r.(\overline{B}_i \sqcap D) \mid i \leq n \text{ and } D \in S_i[C]\}$. Let M be the set of all left-hand sides of concept inclusions 3-11. Now the tree UCQ-rewriting is

$$\varphi = \bigvee_{i \leq 2^{n+1}} \bigvee_{C \in M \cup \{A_0\}} \bigvee_{D \in S_i[C]} D.$$

Clearly, φ is a tree-UCQ of depth exceeding 2^n. It remains to note that every tree-UCQ-rewriting of A_0 under $\mathcal{T}$ must comprise a concept C of depth at least 2^n. Let $X_1, \dots, X_m$, $m = 2^{n+1} - n$, be the sequence of concept names from the set $\{B_0, \dots, B_{n-1}, \overline{B}_0, \dots, \overline{B}_{n-1}\}$ that represent the counter sequence $0, 1, \dots, 2^n - 1$, let $D_0 := A_0$ and $D_{i+1} := B_{i+1} \sqcap \exists r.D_i$ for all $i < 2^n - 1$. Then every tree-UCQ-rewriting of A_0 under $\mathcal{T}$ must comprise the tree-UCQ D_{2^n-1}, which is of depth 2^n. ❑

C From CQs to IQs

A CQ q is *tree-shaped* if the directed graph $(\mathsf{var}(q), \{(x,y) \mid r(x,y) \in q\})$ is a tree and $r(x,y), s(x,y) \in q$ implies $r = s$. Every tree-shaped CQ q can be seen as an $\mathcal{EL}$-concept in the obvious way (see preliminaries). Let $q, \widehat{q}$ be CQs. We say that q' is obtained from q by *fork elimination* if q' can be obtained from q by choosing $r(x_1,y), r(x_2,y) \in q$ with $x_1 \neq x_2$ quantified variables and then identifying x_1 and x_2; we say that q' is a *fork rewriting* of q if q' is obtained from q by zero or more fork eliminations (not necessarily exhaustive).

Let q be a CQ and $x \in \mathsf{var}(q)$. Let $\mathsf{reach}_q(x)$ be the smallest set of variables such that $x \in \mathsf{reach}_q(x)$ and whenever $r(y,z) \in q$ and $y \in \mathsf{reach}_q(x)$, then $z \in \mathsf{reach}_q(x)$. We say that x *is the root of a tree subquery* if

1. $q|_{\mathsf{reach}_q(x)}$, the restriction of q to variables in $\mathsf{reach}_q(x)$, is tree-shaped and contains no answer variable;
2. $r(z,y) \in q$, $y \in \mathsf{reach}_q(x)$, and $y \neq x$ implies $z \in \mathsf{reach}_q(x)$.

The *tree in* q *at* x is the subquery of q that consists of

1. all atoms $A(x)$;
2. all atoms $r(x,y)$ and atoms in $q|_{\mathsf{reach}_q(y)}$ such that y is the root of a tree subquery and $s(x',y) \in q$ implies $x = x'$ and $r = s$.

We use $\mathsf{trees}(q)$ to denote the set of all pairs (x, C) such that $x \in \mathsf{var}(q)$ and C is the tree in q at x, viewed as an $\mathcal{EL}$-concept. We write $q \setminus C$ to denote q with the tree C in q at x removed (i.e., all atoms in C are removed; x is thus removed iff there are no atoms in q outside C that mention it).

For each $x \in \mathsf{var}(q)$, we use $\mathcal{T}_{q,x}$ to denote the TBox that consists of the following CIs:

- $A_q \sqsubseteq A_x$
- $A_x \sqsubseteq A$ for all $A(x) \in q$;
- $A_x \sqsubseteq \exists r.A_y$ for all $r(x,y) \in q$.

Theorem 3. *Let $\mathcal{T}$ be a TBox and q a CQ. Then q is FO-rewritable under $\mathcal{T}$ iff for every fork rewriting q' of q and all $(x,C) \in \mathsf{trees}(q')$, $C \sqcap A_{q' \setminus C}$ is FO-rewritable under $\mathcal{T} \cup \mathcal{T}_{q' \setminus C}$.*

Proof. "if". Let q' be a fork rewriting of q. For every $(x,C) \in \mathsf{trees}(q')$, we can choose an FO-rewriting $\psi_C(x)$ of $C \sqcap A_{q' \setminus C}$ under $\mathcal{T} \cup \mathcal{T}_{q' \setminus C}$. Moreover, we can w.l.o.g. assume $\psi_C(x)$ to be a UCQ [**?**,**?**]. Let $\varphi_C(x)$ be obtained from $\psi_C(x)$ by dropping in every CQ each atom of the form $A_q(x)$ and $A_y(x)$, that is, each atom that involves one of the newly introduced concept names. Define the UCQ $\varphi_{q'}(\mathbf{x})$ as the disjunction over all CQs that can be obtained by choosing a CQ $q_C(x)$ from $\varphi_C(x)$ and then taking the union of

- the CQ q'_0 that can be obtained from q' by removing the tree C at x, for all $(x,C) \in \mathsf{trees}(q')$;

– the chosen CQ $q_C(x)$, for each $(x, C) \in \mathsf{trees}(q')$.

The answer variables in (each CQ in) $\varphi_{q'}(\mathbf{x})$ are the answer variables in q (equivalently q'). Let $\varphi(\mathbf{x})$ be the disjunction over all $\varphi_{q'}(\mathbf{x})$, q' a fork rewriting of q. It remains to show that $\varphi_{q'}(\mathbf{x})$ is an FO-rewriting of q under $\mathcal{T}$.

To this end, first assume that

"only if". DOES NOT GO THROUGH!! Let $q_1 \vee \cdots \vee q_n$ be a UCQ-rewriting of q under $\mathcal{T}$, q' a fork rewriting of q, and $(x, C_0) \in \mathsf{trees}(q')$. Assume to the contrary of what is to be shown that $C_0' := C_0 \sqcap A_{q' \setminus C}$ is not FO-rewritable under $\mathcal{T}' := \mathcal{T} \cup \mathcal{T}_{q' \setminus C, x}$. We show that, then, q is not FO-rewritable under $\mathcal{T}$, obtaining a contradiction. Since C_0' is not FO-rewritable under $\mathcal{T}'$, for every $k > 0$ there is a tree-shaped ABox $\mathcal{A}_k$ with root a such that $\mathcal{A}_k, \mathcal{T}' \models C_0'(a)$ and $\mathcal{A}_k|_k^-, \mathcal{T}' \not\models C_0'(a)$. Let $\mathcal{A}$ be the query $q' \setminus C$ viewed as an ABox and for all $k > 0$, let $\mathcal{A}_k'$ be obtained by taking the union of $\mathcal{A}_k$ and $\mathcal{A}$, identifying the root of $\mathcal{A}_k$ with the individual x in $\mathcal{A}$ (i.e., the root of C in q'). We show that

1. $\mathcal{A}_k', \mathcal{T} \models q(\mathbf{x})$[4] and
2. $\mathcal{A}_k'|_k^-, \mathcal{T} \not\models q(\mathbf{x})$

where $\mathbf{x}$ denotes the answer variables of q, viewed as individual names. It then remains to derive from Points 1 and 2 that q is not FO-rewritable under $\mathcal{T}$. This should be possible by following the proof in our IJCAI paper.

For Point 1, take any model $\mathcal{I}$ of $\mathcal{A}_k'$ and $\mathcal{T}$. We have to show that $\mathcal{I} \models q(\mathbf{x})$. Since $\mathcal{A}_k, \mathcal{T}' \models C_0'(a)$ and by construction of $\mathcal{A}_k'$, we have $\mathcal{A}_k', \mathcal{T}' \models C_0'(x)$. Thus there is a homomorphism h_1 from C to $\mathcal{I}$ that takes the root of C_0' to x.[5] Moreover, there is obviously a homomorphism h_2 from $q' \setminus C$ to $\mathcal{A}$ that takes every variable to itself, thus to $\mathcal{A}_k'$ and to $\mathcal{I}$. It is now easy to combine h_1 and h_2 into a homomorphism from q' to $\mathcal{I}$ that takes all variables in $q' \setminus C$, including the answer variables, to itself. Thus $\mathcal{I} \models q'(\mathbf{x})$, implying $\mathcal{I} \models q(\mathbf{x})$.

For Point 2, it suffices to show that the canonical model $\mathcal{I}$ of $\mathcal{A}_k'|_k^-$ and $\mathcal{T}$ satisfies $\mathcal{I} \not\models q(\mathbf{x})$. From $\mathcal{A}_k|_k^-, \mathcal{T}' \not\models C_0'(x)$, we obtain that the canonical model $\mathcal{J}$ of $\mathcal{A}_k|_k^-$ and $\mathcal{T}'$ satisfies $\mathcal{J} \not\models C_0'(x)$. Now consider the syntactic, rule-based construction of $\mathcal{I}$ as a sequence of interpretations $\mathcal{A}_k'|_k^- = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_n = \mathcal{I}$ and show by induction on i that for all $d \in \Delta^{\mathcal{J}}$ and $\mathcal{EL}$-concepts C, $d \in C^{\mathcal{I}_i}$ implies $d \in C^{\mathcal{J}}$. To do this, we use the construction of $\mathcal{T}'$, which internalizes the query details needed. ❑

Counterexample. Query is $q = t(x, y) \wedge t(y, x) \wedge \wedge r(y, z)$. TBox $\mathcal{T}$ is

$$\exists s.\exists r.\top \sqsubseteq \exists r.\top$$
$$\exists t.\exists s.\top \sqsubseteq \exists r.\top$$

The only fork rewriting is the query itself. Problem: $\exists r.\top \sqcap X$ is not FO-rewritable under $\mathcal{T} \cup \{X \sqsubseteq \exists t.X\}$, but q is FO-rewritable into a UCQ with two disjuncts: q itself and q with the r-edge replaced with an s-edge.

[4] this needs a proper definition; what I mean is: cut in the $\mathcal{A}_k$ part as for $\mathcal{A}_k$ itself, leaving the $\mathcal{A}$ part untouched.

[5] SNA for now for simplicity

This does of course not provide a polytime reduction. But I believe that the Turing reduction could be quite practical. To start with, realistic queries should not contain a large number of forks. Moreover, in practice one should eliminate forks only when really necessary. Something like this:

- iterate over all variables x in the query; check "at each variable for FO-rewritability"; this is motivated by the fact that non-FO-rewritability comes from "propagating around concepts", which are formulas with one free variable.
- start with the concept C_x collecting all concept names at x and all successors of x that are the root of a tree subquery (initially, without any fork elimination); check for FO-rewritability in the "CQ-context", as above using the TBox $\mathcal{T}_{q,x}$.
- if the outcome is "not FO-rewritable", then this is (hopefully) sound;
- otherwise, we can eliminate forks that have their "join" at a node y reachable from x and whose elimination results in C_x to change in a non-equivalent way (relative to $\mathcal{T}$?);
- we have a choice as to which fork we eliminate; the choice seems 'real', that is, in the worst case, we have to go through all possible combinations of joining and not joining the relavant forks. However, a bit of optimization should be possible and, anyway, maybe its not so bad in practical cases;
- in particular: if elimination of fork f_i results in C_x to change to C_{x_i}, $i \in \{1, 2\}$, and $C_{x_1} \sqsubseteq C_{x_2}$ (relative to $\mathcal{T}$?), then first try eliminating f_2 and only then (additionally) f_1;
- Don't forget that, in the end, we want to *compute* a concrete rewriting.