Mixing Materialization and Query Rewriting for Existential Rules

Michaël Thomazo and **Sebastian Rudolph**[1]

Abstract. Ontology-Based Data Access (OBDA) is a recent paradigm aiming at enhancing data access by taking ontological knowledge into account. When using existential rules as ontological language, query answering is an undecidable problem, whence numerous decidable classes of ontologies have been defined, ranging from classes with very good computational complexities (AC_0 in data complexity) to classes with much larger expressivity. However, actually implementable algorithms have been proposed only for very restricted classes (typically those coinciding with lightweight description logics). The aim of this paper is to show how to deal with more expressive ontologies by proposing an algorithm that performs both materialization and rewriting and is applicable for a significant generalization of lightweight description logics. To this end, we first modify an existing algorithm previously proposed for a very generic class of rules, namely greedy bounded treewidth sets of rules. We then exhibit a special case, called pattern oblivious rule sets, which significantly generalizes the $\mathcal{ELH}^{dr}$ description logic, which underlies the OWL 2 EL ontology standard, while keeping the beneficial worst-case computational complexity. We last define a subclass of pattern oblivious rules that is recognizable in polynomial time.

1 Ontology-Based Data Access

In the last few years, a novel paradigm for data querying has become increasingly popular in the knowledge representation and reasoning community as well as in the database community. This paradigm is called Ontology-Based Data Access (OBDA). The key idea is to use an ontology to enrich data with domain knowledge, enabling semantic querying. Current research is mainly focusing on conjunctive queries, which are the basic queries in the database community. The considered decision problem is then formalized as follows: letting F be some data (represented as a set of ground atoms and possibly stored in a relational database), $\mathcal{O}$ an ontology and q a query, does $F \cup \mathcal{O} \models q$ hold?

Depending on the ontology, conjunctive query answering under an ontology can range from undecidable down to AC_0 data complexity (which is the same as conjunctive query answering without any ontology). An intense research effort aimed at defining classes of ontologies for which the conjunctive query answering problem is decidable (or even tractable) has thus taken place, resulting in a comprehensive and diversified zoo of decidable classes.

In this research effort, two different ontology representation paradigms have been intensely studied: Description Logics [4] and existential rules [5], also known as Datalog+/- [7] or tuple-generating dependencies (TGDs) in databases [1]. In Description Logics (DLs), current research is focusing on so-called lightweight DLs, most notably from the $\mathcal{EL}$ [3] and the DL-Lite [8] families. They provide the logical bases of the tractable profiles OWL 2 EL and OWL 2 QL, respectively, of the OWL ontology language [17]. In existential rules, considered classes are usually more expressive, but also do not have as good computational properties as lightweight description logics.

A first approach to design efficient algorithms for OBDA is that of pure query rewriting. The principle is to use the ontology in order to reformulate a query that can be directly evaluated against the original database, which allows (in theory) to make use of good performance of database management systems. This approach is in particular applicable for first-order rewritable ontologies [2, 18, 10, 9, 24, 12, 20] (possibly using Datalog rewritings [11]), but also for $\mathcal{EL}$ [19]. An already known weakness of these approaches is the problem of efficiently evaluating the obtained rewritings, in particular when facing huge unions of conjunctive queries.

Another trend of research allows to overcome this drawback by materializing (part of) the entailed facts. The most naive approach would be to materialize all the entailed facts, but this is not always possible, since there could be infinitely many. Nonetheless, it is in some case possible to modify the data, and to rewrite the query in such a way that when evaluated against the modified data, it yields sound and complete answers. Such an approach, called a *combined approach*, has been applied to DL-Lite and to $\mathcal{ELH}^{dr}$ [16, 13, 14, 15]. However, current combined approach algorithms are tailored towards lightweight description logics only.

The aim of the current paper is to overcome this shortcoming, by providing such a mixed approach (both modifying the data and the query) that is able to deal with ontologies whose expressivity significantly exceeds that of lightweight description logics. The contribution of the present paper is threefold:

- First, we consider the very expressive class of *greedy bounded treewidth sets* [6]. We argue that the known [22] worst-case optimal algorithm is not efficiently implementable, due to an ad-hoc querying operation. We thus propose to replace this operation by the evaluation of a Datalog program, whose size is polynomial in a parameter of the original algorithm, namely the number of so-called *patterns*. While this parameter is high in the worst-case, one can expect it to be small in practical cases. Given an efficient Datalog solver, that would enable our algorithm to work efficiently even on large databases.
- Second, we define an algorithmically simple class of rules by "reverse engineering": we look for expressive classes of rules that ensure that the number of relevant patterns is polynomial. We identify such a class which we call *pattern oblivious* rule sets, which has nice computational properties: query answering is PTIME complete in data complexity and NP-complete in combined com-

[1] TU Dresden, Germany, email: firstname.lastname@tu-dresden.de

plexity under mild restrictions.
- Last, we study the computational complexity of recognizing pattern oblivious rules. We show that it is hard for the second level of the polynomial hierarchy, and thus propose another class of rules, namely *forward-only rules*, that is a particular case of pattern oblivious rules. We show that under mild assumptions, forward-only rules are recognizable in polynomial time.

2 Preliminaries

An *atom* is of the form $p(t_1, \ldots, t_k)$ where p is a predicate of arity k, and the t_i are *terms*, *i.e.*, variables or constants. A *fact* (resp. a *Boolean conjunctive query*) is an existentially closed conjunction of atoms. In this paper, we consider only Boolean queries for the sake of simplicity, but the same techniques can also be applied to non-Boolean queries. Given an atom or a set of atoms A, we denote by $vars(A)$ and $terms(A)$ its set of variables and of terms, respectively. Given two facts F and Q, a *homomorphism* π from Q to F is a mapping from $vars(Q)$ to $terms(F)$ such that $\pi(Q) \subseteq F$. An *existential rule* is a formula $R = \forall \mathbf{x} \forall \mathbf{y}(B[\mathbf{x}, \mathbf{y}] \rightarrow (\exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]))$ where $B = body(R)$ and $H = head(R)$ are conjunctions of atoms, called the *body* and the *head* of R, respectively. The *frontier* of R, denoted by $fr(R)$, is the set of variables $vars(B) \cap vars(H) = \mathbf{y}$. Given a fact F and a set of rules $\mathcal{R}$, we denote by $\mathcal{C}$ the set of constants that appear either in F or in a rule of $\mathcal{R}$. A rule R is *applicable* to a fact F if there is a homomorphism π from $body(R)$ to F; the result of the *application of R to F w.r.t.* π is a fact $\alpha(F, R, \pi) = F \cup \pi^{safe}(head(R))$ where π^{safe} is a substitution applied to $head(R)$, which replaces each $x \in fr(R)$ with $\pi(x)$, and each other variable with a fresh variable. The fusion of the frontier σ_π induced by π is a function from $fr(R)$ to $fr(R) \cup \mathcal{C}$, such that $\sigma_\pi(x) = \sigma_\pi(y)$ if and only if $\pi(x) = \pi(y)$, and $\sigma_\pi(x) = a$ if and only if $p(x) = a$ for any constant a. We assume this fusion to be uniquely defined (for instance, by using a fixed order on the variables). A fusion of the frontier of a rule is a fusion of the frontier induced by some π. An *$\mathcal{R}$-derivation of F* is a finite sequence $F = F_0, F_1, \ldots, F_k = F'$ such that for all $i \in \{1, \ldots, k\}$, there are a rule $R \in \mathcal{R}$ and a homomorphism π from $body(R)$ to F_{i-1} with $F_i = \alpha(F_{i-1}, R, \pi)$. F' is the *result* of the derivation. An *extension* of a derivation $S = F_0, \ldots, F_k$ is a derivation $S' = F'_0, \ldots, F'_n$, with $n > k$ and for all i smaller than k, $F'_i = F_i$. Classically, a Boolean query q is entailed by F and $\mathcal{R}$ if and only if there exists an $\mathcal{R}$-derivation S of F such that q is entailed by the result of S. Proofs are omitted and can be found in [23].

3 On Greedy Bounded Treewidth Sets

The main focus of this paper is the class of greedy bounded treewidth sets [6] and some of its subclasses. The definition of that class relies on the notion of a *greedy derivation*.

Definition 1 (Greedy Derivation) *An $\mathcal{R}$-derivation $(F_0 = F), \ldots, F_k$ is said to be* greedy *if, for all i with $0 < i \leq k$, there is a $j < i$ such that $\pi_i(fr(R_i)) \subseteq vars(A_j) \cup vars(F_0) \cup \mathcal{C}$, where $A_j = \pi_j^{safe}(head(R_j))$.*

A set of rules $\mathcal{R}$ is a greedy bounded treewidth set (*gbts*) if for any fact F, any $\mathcal{R}$-derivation is greedy. The *gbts* class generalizes in particular lightweight description logics, as well as guarded existential rules and their main known generalizations, as well as plain Datalog. A greedy derivation can be associated to a structure called derivation tree, which is a tree decomposition of the primal graph of its result. This derivation tree can be built in a greedy way as made formally precise in the following definition.

Definition 2 (Derivation Tree) *Let $S = (F_0 = F), \ldots, F_k$ be a greedy derivation. The* derivation tree *assigned to S, written: $DT(S)$, is a tree $\mathfrak{T}$ with nodes $\mathcal{B} = \{B_0, \ldots, B_k\}$ (also called* bags*) and two functions $terms : \mathcal{B} \rightarrow 2^{terms(F_k)}$ and $atoms : \mathcal{B} \rightarrow 2^{F_k}$, defined as follows:*

1. *Let $T_0 = vars(F) \cup \mathcal{C}$. The root of the tree is B_0 with $terms(B_0) = T_0$ and $atoms(B_0) = atoms(F)$.*
2. *For $0 < i \leq k$, let R_{i-1} be the rule applied according to homomorphism π_{i-1} to produce F_i; then $terms(B_i) = vars(A_{i-1}) \cup T_0$ and $atoms(B_i) = atoms(A_{i-1})$, where $A_{i-1} = \pi_{i-1}^{safe}(head(R_{i-1}))$. The parent of B_i is the node B_j for which j is the smallest integer where $\pi_{i-1}(fr(R_{i-1})) \subseteq terms(B_j)$.*

Note that, in the second point of the definition, there is at least one j with $\pi_{i-1}(fr(R_{i-1})) \subseteq terms(X_j)$ because S is greedy. Moreover, we always choose the smallest such j, which means that we link the new bag "as high as possible" in the tree.

We now present formal tools to describe bags that are "similar". This is done in particular using the notion of *equivalence function*.

Definition 3 (Equivalence Function) *Let F be a fact and $\mathcal{R}$ be a gbts. Let $\mathcal{P}$ be a set of labels, called* patterns*, partially ordered using a relation $\sqsubseteq$. An* equivalence function *f for F and $\mathcal{R}$ associates an $\mathcal{R}$-derivation S and a bag B of $DT(S)$ with a pattern, in such a way that if S' is an extension of S, then $f(S, B) \sqsubseteq f(S', B)$.*

The term "pattern" is thus used to denote elements of the range of an equivalence function. We will use two different notions of patterns in Section 4 and 5. Let us first introduce the *structure function*, that is central in our development.

Definition 4 (Structure function) *Let F be a fact and $\mathcal{R}$ be a set of rules. The* structure function *$f_\mathcal{R}$ associates any derivation S and any bag B of $DT(S)$ with (R_B, σ_B) where R_B is the rule that created B (by applying π_B), and σ_B the fusion of the frontier induced by π_B.*

An important property of the structure function is that there is a canonical bijection between two bags of a derivation tree whose images by the structure function are equal. If B and B' are such bags, we denote by $\psi_{B \rightarrow B'}$ that bijection. We also define a canonical representative, which can by definition be obtained from any bag B of that class by applying ψ_B to its terms and atoms. We are also interested in how bags are *linked* together, hence Definition 5.

Definition 5 (Link) *Let F be a fact, $\mathcal{R}$ be a set of rules, S be a greedy $\mathcal{R}$-derivation of F. Let B and B' be two bags of $DT(S)$ such that B' is a child of B. The induced link λ between B' and B is a function from $\psi_{B'}(fr(B'))$ to $\psi_B(terms(B))$, defined by: $\lambda(\psi_{B'}(x)) = \psi_B(x)$.*

We restrict our attention to *correct* equivalence functions.

Definition 6 (Correct Equivalence Function) *Let F be a fact, $\mathcal{R}$ be a* gbts*, and f be an equivalence function for F and $\mathcal{R}$. f is correct if for any derivation S and any pair of bags B_1, B_2 in $DT(S)$:*

1. *if $f(S, B_1) \sqsubseteq f(S, B_2)$, then $f_\mathcal{R}(S, B_1) = f_\mathcal{R}(S, B_2)$;*

2. *if $f(S, B_1) = f(S, B_2)$, then if B_1 admits a child B'_1 with induced link λ, then there exists an extension S' of S such that B_2 admits a child B'_2 with $f(S, B'_1) \sqsubseteq f(S', B'_2)$.*

With this vocabulary, it is shown in [22] that the structure function is not a correct equivalence function, but that one can be constructed by refining the structure function by additionally labeling a bag B by the set of pairs $(G, \psi_B \circ \varphi_{|\varphi^{-1}(\mathit{terms}(B))})$, where G is a subset of a rule body and φ is a homomorphism of G into the result of S. This "mappability-knowledge" is completed by means of a saturation mechanism, that halts because of its monotonicity. This knowledge can then be used to perform querying. However, the proposed solution [22] guesses both a suitable tree decomposition of the query and its mapping to the built representation of the canonical model. These successive guesses make the approach unpractical. Our first aim is to improve this querying mechanism, by re-using a Datalog engine.

Thanks to the first point of Definition 6, one can define a canonical representative for any bag whose image by an equivalence function is a pattern P. In particular, we can associate with it a set of terms (resp. frontier terms, atoms) denoted by $\mathit{terms}(P)$ ($\mathit{fr}(P)$, $\mathit{atoms}(P)$, respectively). Moreover, for any bag B such that $f(B) = P$, there is a bijection ψ_B from $\mathit{terms}(B)$ to $\mathit{terms}(P)$ that is also a bijection (with domain suitably restricted) from $\mathit{fr}(B)$ to $\mathit{fr}(P)$ and an isomorphism between $\mathit{atoms}(B)$ and $\mathit{atoms}(P)$.

Provided with a correct equivalence function (giving rise to a finite number of equivalence classes) one can describe derivation trees thanks to a set of structure rules, which state that any bag of some pattern P has a child of pattern P', that is linked with it in a certain way, provided that enough rule applications have been performed. We first formalize the syntax of such a set of structure rules.

Definition 7 (Structure rules) *Let F be a fact and $\mathcal{R}$ be a* gbts. *Let $\sim$ be a correct equivalence relation and let $\mathcal{P}$ be the corresponding set of patterns. A* structure rule *is a rule of the form (P, λ, P') where $P, P' \in \mathcal{P}$ and λ is a mapping from $\mathit{fr}(P')$ to $\mathit{terms}(P)$ such that $\lambda(\mathit{fr}(P')) \not\subseteq \mathit{fr}(P)$. λ is called a* link *between P' and P.*

We then define the notion of correctness of a set of structure rules.

Definition 8 (Structure rule correctness) *A set $\mathcal{S}$ of structure rules is* correct *with respect to a fact F and a set $\mathcal{R}$ of existential rules if:*

- *for every $(P, \lambda, P') \in \mathcal{S}$, for any $\mathcal{R}$-derivation S of F, for every bag B of pattern P in $\mathrm{DT}(S)$, there exists an extension S' of S such that B has a child B' of pattern P' in $\mathrm{DT}(S')$ that is linked to B via $\psi_B^{-1} \circ \lambda$, and (soundness)*
- *for any derivation S, for any bags B and B' of respective patterns P and P' such that B is a child of B' with induced link λ, then $(P, \psi_B^{-1} \circ \lambda, P')$ belongs to $\mathcal{S}$. (completeness)*

Obviously, a set of structure rules can also be seen as a way to generate facts. *In this paper, we assume that such a set of rules is already computed*. This is a non-trivial task, and the interested reader is invited to consult [21], where so-called creation rules allow to build structure rules. Let us point out that structure rules are a finite representation of the canonical model (also known as chase) of F and $\mathcal{R}$. Moreover, this finite representation is easier to use than F and $\mathcal{R}$, since it provides full (certain) information on each individual as soon as it is introduced. In particular, it encapsulates in the pattern corresponding to the initial fact all the atoms entailed by the knowledge base that have as arguments terms from the initial fact.

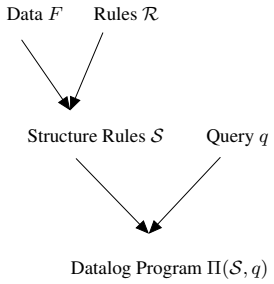

Figure 1. Workflow of the proposed algorithm

4 Datalog Rewriting for Greedy Bounded Treewidth Sets

In this section, we present a rewriting mechanism for *gbts* rules. We first explain, in a high-level fashion, the main ideas of this rewriting operation, then provide a formal presentation of the rewriting. This rewriting mechanism takes as input a set of structure rules $\mathcal{S}$ and a query q. It outputs a Datalog program $\Pi(\mathcal{S}, q)$.

4.1 High-level Presentation of the Rewriting

We design a Datalog program that computes homomorphisms of subsets of the query into the set of atoms contained in the patterns. In other words, we inspect the patterns in order to identify partial query matches. For each pattern P appearing in $\mathcal{S}$, this will be done thanks to a fresh predicate q_P, of arity $j + k$, where j is the number of atoms in the query and k is the number of terms in the query. Intuitively, the first j positions of the atom carry the information about which atoms are mapped by the homomorphism encoded, and the last k positions represent which of the query variables are mapped into the considered pattern (and, in the positive case, to which terms of the pattern they are mapped). As we are interested in partial matches, the homomorphism may not contain every variable of the query in its domain, thus we make use of a special constant symbol ($\square$) to represent the case where the images of some variables are not (yet) specified. Initially, only mappings of single atoms are represented. Mappings of larger parts of the query will be obtained thanks to a rule allowing for combining compatible partial matches into larger ones.

Before introducing this combination rule, let us point out that a homomorphism from (a part of) the query into the ultimate derivation tree may map different atoms into different bags. We account for this thanks to structure rules, using them to propagate information about partial homomorphisms from one pattern to another. In the course of this propagation, it is possible that the image of a term x belonging to the terms of a source pattern does not belong to the target pattern: we thus only specify, using a new special constant symbol ($\times$), that the image of x is already fixed, but is not available in the current bag. This imposes that we cannot choose an(other) image for x anymore.

Information being propagated between different bags, we need to merge different pieces of information. That is, if we know two partial homomorphisms that are compatible, we can infer the existence of a joint homomorphism that maps atoms as mapped by the two homomorphisms. In order to get a rewriting that is polynomial in the query, we make use of the fresh *compatible* predicate, that encodes the compatibility of two terms and the result of their unification.

4.2 Formal Presentation of the Rewriting

We now formally present the Datalog rewriting. We take as input a set of correct structure rules, and a query q. We enumerate the atoms of the query from 1 to j, and the variables of the query from 1 to k.

4.2.1 Initializing Patterns Predicates

Let P be a pattern. We bijectively associate each term of P with a fresh constant by π_P, and for every atom $a(x_1, \ldots, x_k)$ assigned to P, we create an atom $a(\pi_P(x_1), \ldots, \pi_P(x_k))$. We also associate with P a fresh predicate q_P, whose arity equals the number of atoms in q plus the number of variables in q. If a_i is the i^{th} atom of the query, and there is a homomorphism π from a_i to $\mathit{atoms}(P)$, we create the following rule:

$$\rightarrow q_P(e_1, \ldots, e_j, t_1, \ldots, t_k),$$

where:

- $e_\ell = 1$ if $\ell = i$, 0 otherwise;
- $t_\ell = \pi_P(\pi(x_\ell))$, if x_ℓ belongs to the arguments of a_i, $t_\ell = \square$ otherwise.

4.2.2 Propagating Partial Homomorphisms

We now create a predicate *link* that specifies correspondences between terms of a bag and terms of its immediate parent. Let (P, λ, P') be a structure rule. We first create the following two rules:

- $\rightarrow \mathit{link}_{P,\lambda,P'}(\times, \times)$;
- $\rightarrow \mathit{link}_{P,\lambda,P'}(\square, \square)$.

The first one specifies that a term that has been mapped, but whose image has been forgotten remains in that case when we propagate the information in a new pattern. The second rule specifies that a term that has not yet been mapped remains unmapped. Then, for any term z such that $\lambda(z) = y$, we create the following rule:

$$\rightarrow \mathit{link}_{P,\lambda,P'}(\pi_{P'}(z), \pi_P(y)),$$

For any constant x that does not belong to the domain of λ, we create a new rule:

$$\rightarrow \mathit{link}_{P,\lambda,P'}(\pi_{P'}(x), \times).$$

The propagation rule is then:

$$q_{P'}(x_1, \ldots, x_j, y_1, \ldots, y_k) \wedge \bigwedge_{1 \leq i \leq k} \mathit{link}_{P,\lambda,P'}(y_i, y'_i)$$

$$\rightarrow q_P(x_1, \ldots, x_j, y'_1, \ldots, y'_k).$$

4.2.3 Combining Partial Homomorphisms

To combine partial homomorphisms with a Datalog program, we use an auxiliary ternary predicate, *compatible*. First, on special symbols 0 and 1, it states that if at least one of the homomorphisms maps an atom, then the combined homomorphism does so as well.

- $\mathit{compatible}(0, 0, 0)$
- $\mathit{compatible}(0, 1, 1)$
- $\mathit{compatible}(1, 0, 1)$
- $\mathit{compatible}(1, 1, 1)$

Then, the predicate also checks that images of the same variable are not differently defined in both homomorphisms.

- $\mathit{compatible}(x, x, x)$, for any $x \neq \times$
- $\mathit{compatible}(x, \square, x)$, for any x (including $\times$)
- $\mathit{compatible}(\square, x, x)$, for any x (including $\times$)

We thus create one combination rule per pattern:

$$q_P(x_1, \ldots, x_{j+k}) \wedge q_P(x'_1, \ldots, x'_{j+k}) \wedge$$

$$\bigwedge_{1 \leq i \leq j+k} \mathit{compatible}(x_i, x'_i, x''_i) \rightarrow q_P(x''_1, \ldots, x''_k).$$

Last, we introduce a predicate goal, with a rule per pattern:

$$q_P(1, \ldots, 1, x_1, \ldots, x_k) \rightarrow \mathit{goal}.$$

4.3 Properties of the Rewriting

From Property 1 follows the correctness of $\Pi(\mathcal{S}, q)$, *i.e.*, q is entailed by F and $\mathcal{R}$ if and only if *goal* is entailed by the Datalog rewriting designed in the previous section. Let q be a query of j (ordered) atoms. Let $b = (b_1, \ldots, b_j)$ is a tuple of size j whose elements are either 0 or 1. We denote by q_b the subset of q that contains exactly the i^{th} atom of q, for all i such that b_i is equal to 1. For an atom $a = q_P(b_1, \ldots, b_j, y_1, \ldots, y_k)$, we define $q_a = q_{(b_1, \ldots, b_k)}$. We also define π_a as the function $\{x_i \mapsto y_i \mid y_i \notin \{\times, \square\}\}$, that is, π_a maps exactly those terms x_i of q to their respective y_i for which y_i is different from $\times$ and $\square$. We can now express Property 1.

Property 1 (Correctness of the Rewriting) *Let $\mathcal{S}$ be a set of structure rules, and q be a conjunctive query. It holds that $a = q_P(b_1, \ldots, b_j, y_1, \ldots, y_k)$ is entailed by $\Pi(\mathcal{S}, q)$ if and only if for any bag B of pattern P, there exists a homomorphism π from q_a to the atoms associated with the tree generated from B by $\mathcal{S}$ such that $\Psi_B \circ \pi_{|\mathit{dom}(\pi_a)} = \pi_a$.*

Property 2 (Size of the Rewriting) $\Pi(\mathcal{S}, q)$ *contains $\mathcal{O}(p \cdot |q| \cdot t^k + p^2 \cdot t^f)$ rules, where p is the number of patterns, t is the maximum number of terms associated with a pattern, k is the maximum arity of a predicate and f is the maximum size of a frontier of a rule.*

5 Pattern Oblivious Sets of Rules

The rewriting presented in the previous section is polynomial in the number of patterns. Recall that patterns are elements of the range of an equivalence function. A straightforward way to ensure that it is also polynomial in $\mathcal{R}$, F and q is thus to ensure that there exists a correct equivalence function of range of polynomial size. This is not possible in the general case of *gbts*, and we thus reverse-engineer by instantiating the patterns and then considering the adequate rule sets.

5.1 Definition and Links with other Known Classes

In this section, we focus on the structure function, and a pattern will thus be a pair (R_P, σ_P) where R_P is a rule and σ_P a fusion of its frontier. Moreover, two patterns are incomparable if they are distinct. We still associate a pattern with terms, frontier, and atoms, by taking the image of the head of R_P by σ_P. We then define pattern oblivious rulesets in a straightforward (but not constructive) way as follows.

Definition 9 (Pattern Obliviousness) *A set of rules $\mathcal{R}$ is* pattern oblivious *if it is* gbts *and if for any fact F, the structure function is a correct equivalence function.*

When having a polynomial number of patterns, the rewriting proposed in the previous section allows to derive a non-deterministic polynomial algorithm for conjunctive query answering, which is polynomial in data complexity. Together with lower bounds coming from the coverage properties of $\mathcal{EL}$ stated below, this allows to state the following complexity results.

Property 3 (Complexity of Conjunctive Query Answering) *Conjunctive query answering under pattern oblivious sets of rules has* PTIME*-complete data complexity. Its combined complexity is* NP*-complete if the frontier size and the arity are bounded.*

Before turning to the complexity of the recognition of pattern oblivious rules, let us point out that this existential rule fragment covers lightweight description logics classically used for OBDA.

Property 4 (Covering of $\mathcal{ELH}^{dr}$) *Let $\mathcal{R}$ be the canonical translation of an $\mathcal{ELH}^{dr}$ ontology into first-order logic. $\mathcal{R}$ is a pattern oblivious set of rules.*

Linear rules are existential rules where the body contains at most one atom. Linear rules (and thus DL-Lite$_\mathcal{A}$) are pattern oblivious.

Property 5 (Covering of Linear Rules) *Let $\mathcal{R}$ be a set of linear rules. Then, $\mathcal{R}$ is pattern oblivious.*

In summary, pattern oblivious rules are strictly more expressive than $\mathcal{ELH}^{dr}$ and DL-Lite$_\mathcal{A}$ without complexity increase compared to $\mathcal{ELH}^{dr}$. This expressivity increase also exists while keeping the typical requirement of DLs relative to arities and acyclic rules.

Example 1 *Let us consider consider the following translation of an $\mathcal{ELI}$ ontology: $\mathcal{R}_{\mathcal{ELI}} = \{R_1 = p(x) \rightarrow r(x,y), R_2 = r(x,y) \wedge h(x) \rightarrow r(y,z)\}$. Let us consider $F = \{p(a), h(a), p(b)\}$. R_1 is applicable by mapping its frontier either to a or b, creating two bags in the associated derivation tree, B_a (resp. B_b) with associated atom $r(a, y_a)$ (resp. $r(b, y_b)$). B_a and B_b have the same image by the structure function. However, R_2 is applicable only by mapping its frontier to y_a, and no other rules are applicable. This shows that the structure function is not a correct equivalence function, and thus $\mathcal{R}_{\mathcal{ELI}}$ is not pattern oblivious.*

$\mathcal{R}'_{\mathcal{ELI}} = \{R_1 = p(x) \rightarrow s(x,y) \wedge s(z,y) \wedge h(z), R_2 = s(x,y) \wedge h(x) \rightarrow s(x,z) \wedge p(z)\}$ is expressible neither in $\mathcal{ELH}^{dr}$ nor in DL-Lite$_\mathcal{A}$. It is however pattern oblivious, since any bag created by R_1 has a similar child created by R_2 mapping its frontier to the fresh instantiation of z, and a similarly for bags created by R_2.

Last, let us point out that pattern-oblivious rules are syntactically incomparable with other known class of existential rules that admit polynomial data complexity, such as guarded rules [7]. Example 1 already shows that guarded rules are not necessarily pattern oblivious. For the converse, one can consider $\{r(x,y) \wedge s(y,z) \rightarrow r(z,t) \wedge s(t,u)\}$.

5.2 Complexity of the Recognition Problem

Unfortunately, it happens that deciding if a set of rules is pattern oblivious is a complex problem.

Property 6 *Deciding if a given set of rules is pattern-oblivious is a Π_2^p-hard problem.*

We thus define *forward-only* rules, which are a specific case of pattern oblivious rules, and show that forward-only rules are recognizable in polynomial time, provided the size of their bodies is bounded. This assumption is natural in practical cases, and it is worth to note that considering rules with bodies of bounded size does not ensure decidability of the conjunctive query answering problem if no further constraints are considered.

Since recognizing pattern obliviousness is hard, we define a more restricted class of rules, that implies pattern obliviousness, and that can be recognized in polynomial time.

Definition 10 (Forward-only Sets of Rules) *Let $\mathcal{R}$ be a set of existential rules. $\mathcal{R}$ is* forward-only *if it is* gbts *and if it holds that for any fact F, any $\mathcal{R}$-derivation S of F, any bag B of $DT(S)$, any rule R in $\mathcal{R}$, if π is a homomorphism of $\mathsf{body}(R)$ into F_k such that the image of the frontier of R is included in terms of B but not of its parent, then $\pi(\mathsf{body}(R))$ is a subset of the atoms associated with the subtree of $DT(S)$ rooted in B.*

Property 7 *Let $\mathcal{R}$ be a set of existential rules. If $\mathcal{R}$ is forward-only, then $\mathcal{R}$ is pattern oblivious.*

We now focus on the recognizability of forward-only rules. To prove this, we associate with any set of existential rules a set of structure rules that represents which rule may be applied (and in which way), provided that we restrict ourselves to rule applications that satisfy the conditions of Definition 10.

Definition 11 (Entailment of a Structure Rule) *Let $\mathcal{R}$ be a set of rules, $\mathcal{S}$ be a (possibly empty) set of structure rules, P and P' two weak equivalence patterns (associated with (R_P, σ_P) and $(R_{P'}, \sigma_{P'})$) and (P, λ, P') be a structure rule. We say that (P, λ, P') is* entailed by $\mathcal{S}$ and $\mathcal{R}$ *if at least one of the following three conditions holds:*

- $(P, \lambda, P') \in \mathcal{S}$;
- *there is a homomorphism π from $\mathsf{body}(R_{P'})$ to $\sigma_P(\mathsf{head}(R_P))$ such that for any $x \in \mathsf{fr}(R_{P'})$, $\pi(x) = \lambda(x)$;*
- *the following two conditions hold:*
  - $\Pi(\mathcal{S}, \mathsf{body}(R_{P'})) \models q_P(1, \ldots, 1, y_1, \ldots, y_k)$;
  - $\lambda = \pi_{q_P(1,\ldots,1,y_1,\ldots,y_k)|\mathsf{fr}(R_{P'})}$,

  where $\pi_{q_P(1,\ldots,1,y_1,\ldots,y_k)}$ is defined as in Property 1.

Since there is a finite number of structure rules and that $\mathcal{S}$ and $\mathcal{R}$ entail $\mathcal{S}$, Definition 12 is valid.

Definition 12 (Oblivious Structure Rules) *Let $\mathcal{R}$ be a set of rules. The set of* oblivious structure rules *of $\mathcal{R}$ is obtained as follows. Let $\mathcal{S}_0 = \emptyset$. For any $i > 1$, $\mathcal{S}_i$ is equal to the set of structure rules that are entailed by $\mathcal{R}$ and $\mathcal{S}_{i-1}$. The set of oblivious structure rules of $\mathcal{R}$, denoted by $\mathcal{S}_\mathcal{R}$, is the first $\mathcal{S}_i$ such that $\mathcal{S}_i = \mathcal{S}_{i+1}$. For any $i \geq 1$, a structure rule that belongs to $\mathcal{S}_i \setminus \mathcal{S}_{i-1}$ is said to have* rank i.

In the following we state two technical properties of the set of oblivious structure rules of a set of existential rules. First, Property 8 deals with the soundness of the set of oblivious structure rules, while Property 9 focuses on a notion of completeness.

Property 8 *Let $\mathcal{R}$ be a set of existential rules, and $\mathcal{S}_\mathcal{R}$ the set of oblivious structure rules of $\mathcal{R}$. If $(P, \lambda, P') \in \mathcal{S}_\mathcal{R}$, then for any fact F, any greedy $\mathcal{R}$-derivation $F_0, \ldots, F_k$ of F, any bag B of pattern P in $DT(S)$, there exists an extension S' of S that is greedy and is such that B has a child B' of pattern P' and of link $\psi_B^{-1} \circ \lambda$ with B.*

Property 9 *Let $\mathcal{R}$ be a set of existential rules, F be a fact, S be a greedy derivation of $F = F_0, \ldots, F_k$, such that for any $i > 1$, $F_i = \alpha(F_{i-1}, R_i, \pi_i)$. We denote by $B_{p(i)}$ the parent of the bag of $DT(S)$ created by the application of R_i by π_i. We assume that S is such that for any i, π_i maps the body of R_i to the atoms associated with the subtree rooted in $B_{p(i)}$. Let B, B' be bags of $DT(S)$ such that $f_{\mathcal{R}}(S, B) = P$ and $f_{\mathcal{R}}(S, B') = P'$, and that B' is a child of B with corresponding link λ. Then (P, λ, P') belongs to $\mathcal{S}_{\mathcal{R}}$.*

We can now use these two properties to prove the following result.

Property 10 *Let k be a fixed integer. One can decide in polynomial time whether a set of existential rules $\mathcal{R}$ (whose rule bodies have less than k atoms) is a set of forward-only rules.*

We finish this section by providing the reader with an example of pattern-oblivious set of rules that is not forward-only.

Example 2 *Let us consider $\mathcal{R} = \{R_1, R_2, R_3\}$, with $R_1 = p(x) \rightarrow r(x, y)$, $R_2 = r(x, y) \wedge q(x) \rightarrow s(y, z)$ and $R_3 = r(x, y) \rightarrow q(x)$. Let $F = \{p(a), q(a)\}$. R_1 is applicable, creating $r(a, x_1)$. R_2 is then applicable, by mapping its frontier to x_1. However, an atom of the initial fact is used to map the body of R_2, and thus, the image is not included in the subtree rooted in the bag created by the application of R_1. This shows that $\mathcal{R}$ is not forward-only. Nonetheless, $\mathcal{R}$ is pattern oblivious. Given the rule set, it is enough to check that all bags created by the rule R_1 have equivalent children. This is the case, since R_3 allows to add the necessary atom to trigger an application of R_2 whenever R_1 is applied. Thus, the structure function is not a correct equivalence function.*

6 Conclusion and Further Work

In this work, we considered the recently introduced class of greedy bounded treewidth sets of rules and proposed a novel algorithm that aims at taking advantage of database technology. We first improved an already existing algorithm [22], modifying the ad-hoc querying operation developed in it by the evaluation of a Datalog program, with the aim of enabling the use of existing Datalog solvers. Since *gbts* rules are extremely complex, we also identified a large class of rules on which our algorithm is readily applicable and that have good computational properties: conjunctive query answering under such set of rules is NP-complete in combined complexity (when the arity and the frontier size are fixed) and PTIME-complete in data complexity. We also defined one of its subclasses, namely the one of forward-only rules, that is polynomially recognizable under mild assumptions. Moreover, these classes of rules are a significant generalization of lightweight description logics that are the basis of widely used Semantic Web languages ($\mathcal{EL}\mathcal{H}_{\bot}^{dr}$ and DL-Lite$_{\mathcal{A}}$): indeed, even when restricting them further to the typical DL restrictions, they still provide a strict generalization of these lightweight description logics. We believe that the presented classes are a good trade-off between expressivity and complexity of reasoning.

As future work, we are going to implement the proposed algorithm for large subclasses of *gbts*, including in particular pattern oblivious and guarded sets of rules. The practical evaluation of such an algorithm is not straightforward, as rule sets that actually make use of all the allowed features are not available yet. We believe this is due to the fact that appropriate tools to develop and use such ontologies are not available yet (another instance of the well-known chicken-and-egg problem), and not to the fact that the presented features are not useful. We also believe that the oblivious set of structure rules is interesting in its own right, since it could be a useful tool in the study of approximate reasoning, in particular by studying the difference of semantics between an arbitrary rule set and the Datalog program generated from its set of oblivious structure rules.

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison Wesley, 1994.

[2] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati, 'Quonto: Querying ontologies', in *AAAI*, pp. 1670–1671, (2005).

[3] F. Baader, 'Terminological cycles in a description logic with existential restrictions', in *IJCAI*, pp. 325–330, (2003).

[4] *The Description Logic Handbook: Theory, Implementation, and Applications*, eds., F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, Cambridge University Press, second edn., 2007.

[5] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat, 'On Rules with Existential Variables: Walking the Decidability Line', *Artif. Intell.*, **175**(9-10), 1620–1654, (2011).

[6] J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo, 'Walking the complexity lines for generalized guarded existential rules', in *IJCAI*, pp. 712–717, (2011).

[7] A. Calì, G. Gottlob, and T. Lukasiewicz, 'A general datalog-based framework for tractable query answering over ontologies', *J. Web Sem.*, **14**, (2012).

[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, 'Dl-lite: Tractable description logics for ontologies', in *AAAI*, pp. 602–607, (2005).

[9] A. Chortaras, D. Trivela, and G. B. Stamou, 'Optimized query rewriting for OWL 2 QL', in *CADE*, pp. 192–206, (2011).

[10] G. Gottlob, G. Orsi, and A. Pieris, 'Ontological queries: Rewriting and optimization', in *ICDE*, pp. 2–13, (2011).

[11] G. Gottlob and T. Schwentick, 'Rewriting ontological queries into small nonrecursive datalog programs', in *KR*, (2012).

[12] M. König, M. Leclère, M.-L. Mugnier, and M. Thomazo, 'A sound and complete backward chaining algorithm for existential rules', in *RR*, pp. 122–138, (2012).

[13] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev, 'The combined approach to query answering in dl-lite', in *KR*, (2010).

[14] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev, 'The combined approach to ontology-based data access', in *IJCAI*, pp. 2656–2661, (2011).

[15] C. Lutz, I. Seylan, D. Toman, and F. Wolter, 'The combined approach to OBDA: Taming role hierarchies using filters', in *SSWS+HPCSW*, (2012).

[16] C. Lutz, D. Toman, and F. Wolter, 'Conjunctive Query Answering in the Description Logic $\mathcal{EL}$ Using a Relational Database System', in *IJCAI*, pp. 2070–2075, (2009).

[17] *OWL 2 Web Ontology Language: Profiles*, eds., Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz, W3C Recommendation, 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

[18] R. Rosati and A. Almatelli, 'Improving query answering over dl-lite ontologies', in *KR*, (2010).

[19] G. Stefanoni, B. Motik, and I. Horrocks, 'Small datalog query rewritings for EL', in *Description Logics*, (2012).

[20] M. Thomazo, 'Compact rewriting for existential rules', in *IJCAI*, (2013).

[21] M. Thomazo, *Conjunctive Query Answering under Existential Rules – Complexity, Decidability and Algorithms*, Ph.D. dissertation, Université Montpellier II, 2013. Available at http://www.inf.tu-dresden.de/content/institutes/ki/cl/people/data/thomazo-thesis.pdf.

[22] M. Thomazo, J.-F. Baget, M.-L. Mugnier, and S. Rudolph, 'A generic querying algorithm for greedy sets of existential rules', in *KR*, (2012).

[23] M. Thomazo and S. Rudolph. Mixing materialization and query rewriting for existential rules, 2013. Available at http://www.inf.tu-dresden.de/content/institutes/ki/cl/people/data/report-ecai-14-tr.pdf.

[24] T. Venetis, G. Stoilos, and G. B. Stamou, 'Incremental query rewriting for OWL 2 QL', in *Description Logics*, (2012).