On the Complexity of Ontological Reasoning under Disjunctive Existential Rules

Georg Gottlob[1,2,3], Marco Manna[1,4], Michael Morak[1], and Andreas Pieris[1]

[1] Department of Computer Science, University of Oxford, UK
[2] Oxford-Man Institute of Quantitative Finance, University of Oxford, UK
[3] Institute for the Future of Computing, Oxford Martin School, UK
[4] Department of Mathematics, University of Calabria, Italy
{georg.gottlob,michael.morak,andreas.pieris}@cs.ox.ac.uk,
manna@mat.unical.it

Abstract. Ontology-based data access is an emerging yet powerful technology that allows to enhance a classical relational database with an ontology in order to infer new intensional knowledge. Recently, Datalog+/- was introduced with the purpose of providing tractable reasoning algorithms for expressive ontology languages. In this framework, Datalog is extended by features such as existential quantification in rule heads, and at the same time the rule syntax is restricted to guarantee decidability, and also tractability, of relevant reasoning tasks. In this paper, we enrich Datalog even more by allowing not only existential quantification but also disjunction in rule heads, and we investigate the complexity of reasoning under the obtained formalism.

1 Introduction

Ontological reasoning is a fundamental task in the Semantic Web, where the information present in the web is annotated in order to be machine-readable. Semantic Web ontologies are modeled using logical formalisms, such as W3Cs standard OWL 2 DL ontology language[1] which is built on *Description Logics (DLs)* [6]. DLs are decidable fragments of first-order logic based on concepts (classes of objects) and roles (binary relations between concepts). A large corpus of works in DLs has focused on the problems of consistency (whether a knowledge base is consistent or satisfiable), instance checking (whether a certain object is an instance of a concept), and logical entailment (whether a certain constraint is logically implied by an ontology). The last few years, the attention has shifted on the problem of query answering. A notable example is the *DL-Lite* family of DLs [18,30], which forms the OWL 2 QL profile[2] of OWL 2 DL, that offers flexible, natural and expressive languages, and at the same time keeps query answering highly tractable and scalable to large data sets.

Recently, the DL-Lite family (together with other well-known DLs such as the *DLR-Lite* family [17] and $\mathcal{EL}$ [4]) has been embedded into an expressive framework called *Datalog*$^{\pm}$ [14]. The Datalog$^{\pm}$ family has been proposed with the purpose of providing

[1] http://www.w3.org/TR/owl2-overview/
[2] http://www.w3.org/TR/owl2-profiles/

tractable reasoning algorithms for more general ontology languages. These languages are based on Datalog rules[3] that allow for existentially quantified variables in rule heads, in the same fashion as Datalog with *value invention* [11]. Such rules are known as *tuple-generating dependencies (TGDs)* in the database literature [9]. In particular, TGDs are implications between conjunctions of atoms. They essentially say that some tuples in an instance I imply the presence of some other tuples in I. For example, the TGD

$$\forall E \forall D\, \mathit{emp}(E) \wedge \mathit{mgr}(E, D) \rightarrow \exists E'\, \mathit{emp}(E') \wedge \mathit{works}(E', D) \wedge \mathit{reports}(E', E)$$

expresses the following: "if an employee E is the manager of a department D, then there is an employee E' who works in D and reports to E." The absence of value invention, thoroughly discussed in [29], is the main shortcoming of Datalog in modeling ontologies, and even conceptual data formalisms such as UML class diagrams [10] and ER schemata [19]. The addition of value invention in the form of existential quantification in rule heads constitutes a crucial step towards the bridging of the gap between ontology languages and Datalog, and opens new horizons in ontological reasoning.

Unfortunately, the addition of existential quantification in Datalog easily leads to undecidability of the most basic reasoning tasks [9,12]. Currently, an important research direction in the general area of knowledge representation and reasoning is to identify expressive Datalog$^\pm$ formalisms (or, equivalently, classes of TGDs) for which the basic reasoning services are decidable. Moreover, to be able to work with very large data sets, it is desirable not only that reasoning is decidable, but also tractable in data complexity (i.e., in the size of the database). The main syntactic paradigms which guarantee the above desirable properties are *weak-acyclicity* [22], *guardedness* [12], *stickiness* [15] and *shyness* [27]. Several attempts have been conducted towards the identification of even more expressive formalisms, by extending or combining the above classes (see, e.g., [7,16,26,28]).

The aforementioned Datalog-based formalisms, in contrast to expressive DL-based ontology languages, are not powerful enough for nondeterministic reasoning. For instance, in the well-known description logic $\mathcal{ELU}$, that is, $\mathcal{EL}$ extended with disjunction (see, e.g., [5]), it is possible to state simple and natural statements such as "the parent of a father is a grandfather *or* a grandmother" using the following axiom:

$$\exists \mathit{parentOf}.\mathit{father} \sqsubseteq \exists \mathit{grandfatherOf}.\top \sqcup \exists \mathit{grandmotherOf}.\top$$

This axiom can be expressed by a guarded TGD extended with disjunction as follows:

$$\forall X \forall Y\, \mathit{parentOf}(X, Y) \wedge \mathit{father}(Y) \rightarrow \exists Z\, \mathit{gfatherOf}(X, Z) \vee \mathit{gmotherOf}(X, Z)$$

Obviously, to represent such kind of knowledge, we need to extend the existing classes of TGDs by allowing disjunctive statements in the head. *Disjunctive Datalog*, that is, the variant of Datalog where disjunction may appear in rule heads, has been thoroughly investigated by Eiter et al. [21]. The data complexity of reasoning under guarded-based classes of TGDs extended with disjunction has been recently studied in [2].

Research Challenge. It is the precise aim of the current work to better understand the problem of reasoning under *disjunctive TGDs (DTGDs)*, that is, TGDs that allow for

[3] A Datalog program is a set of universally quantified function-free Horn clauses (see, e.g., [1]).

disjunction in their heads, and to investigate the complexity of reasoning under existing classes of TGDs extended with disjunction. In particular, we concentrate on the problem of *atom entailment* defined as follows: given an extensional database D, an ontology Σ constituted by DTGDs, and a relational atom $\underline{a}$, decide whether each model of D and Σ, i.e., each instance that contains D and satisfies Σ, entails $\underline{a}$, denoted $D \cup \Sigma \models \underline{a}$. Notice that the problem of instance checking is a special case of atom entailment.

Ontological reasoning adheres to the standard logical semantics of entailment ($\models$), which denotes entailment under arbitrary, not necessarily finite, models. This implies, in general, that a database and a set of DTGDs admit infinitely many models, where each of them can be of infinite size; in fact, this holds already for TGDs due to the existential quantification. Interestingly, in the case of TGDs, it is always possible to construct the so-called *universal model*, which can be seen as a representative of all the other models, by applying a well-known procedure called *chase* (see, e.g., [20,24]). Roughly, the chase adds new atoms to the given database as dictated by the given TGDs, possibly involving labeled null values as witnesses for the existentially quantified variables, until the final result satisfies all the TGDs. Hence, for reasoning purposes, instead of considering all the models of an ontological theory constituted by TGDs, we can concentrate on the universal model.

The situation changes dramatically if we consider DTGDs. In fact, there is no single model that acts as a representative of all the other models. The notion of universal model has been recently extended to the disjunctive case by introducing *universal model sets* [2]. It was shown that, given an extensional database D and a set Σ of DTGDs, by applying a procedure similar to the chase, one can build a set $P_{D,\Sigma}$ of ground rules, called the *instantiation* of D and Σ. The models of $P_{D,\Sigma}$ constitute the universal model set for D and Σ. Nevertheless, $P_{D,\Sigma}$ admits, in general, infinitely many models, that have to be considered for ontological reasoning purposes. From the above discussion, it is clear that atom entailment (and other reasoning tasks) is becoming even more complicated and challenging if we consider DTGDs instead of TGDs. Obviously, correct and terminating algorithms require methods for reasoning over infinite structures without explicitly building them.

Summary of Contributions. We investigate the complexity of atom entailment under guarded and linear DTGDs. Recall that guarded TGDs is one of the main classes of TGDs that guarantees decidability, and also tractability in data complexity, of reasoning [12]. A TGD is *guarded* if it has an atom in its body that contains all the

Table 1. Complexity of atom entailment under guarded and linear DTGDs

Formalism	Data Complexity	DTGDs of fixed size	Combined Complexity
Guarded DTGDs	**coNP**-complete UB: [8] LB: [2]	**EXPTIME**-complete UB: [23] LB: [12]	**2EXPTIME**-complete UB: [23] LB: [12]
Linear DTGDs	in $\mathbf{AC}_0$ UB: Thm. 3	**PTIME**-complete UB: Thm. 2 LB: Thm. 5	**EXPTIME**-complete UB: Thm. 2 LB: Thm. 4

body-variables, while a TGD is *linear* if it has only one body atom (and thus it is guarded). The complexity results established in this paper are presented in Table 1; we have indicated, in each cell, where to find the corresponding results (UB and LB stand for upper bound and lower bound, respectively). A summary of our contribution follows:

– We show that atom entailment under guarded DTGDs is **coNP**-complete in data complexity, **EXPTIME**-complete in case of DTGDs of fixed size, and **2EXPTIME**-complete in combined complexity, i.e., the complexity calculated by considering, together with the database, also the set of DTGDs as part of the input. The upper bounds are obtained from existing results on satisfiability of logical theories that fall in the guarded fragment of first-order logic [8,23]. In particular, we show that atom entailment under guarded DTGDs can be reduced to the problem of unsatisfiability of guarded first-order theories. The lower bounds are inherited immediately from existing results on atom entailment under guarded DTGDs. We also show that every $\mathcal{ELU}$ ontology [5] can be rewritten into an equivalent set (w.r.t. atom entailment) of guarded DTGDs.
– We show that atom entailment under linear DTGDs is in $\mathbf{AC}_0$ in data complexity[4] (improving the **LOGSPACE** upper bound established in [2]), **PTIME**-complete in the case of DTGDs of fixed size, and **EXPTIME**-complete in combined complexity. These are novel complexity results, and constitute the main contribution of this paper. The $\mathbf{AC}_0$ upper bound is obtained by establishing that atom entailment under linear DTGDs can be reduced to the evaluation of a first-order query over a database [32]. The remaining two upper bounds are obtained by giving an alternating algorithm which runs in logarithmic space if the DTGDs have fixed size, and in polynomial space in general. The lower bounds are established by simulating the behavior of an alternating logarithmic space (resp., polynomial space) Turing machine by means of linear DTGDs.

2 The Framework

In this section, we present background material necessary for this paper. We recall some basics on relational databases, we introduce disjunctive tuple-generating dependencies, and we discuss the problem tackled in this work, i.e., atom entailment.

Technical Definitions. We define the following pairwise disjoint (countably infinite) sets of symbols: a set Γ of *constants* (constituting the "normal" domain of a database), a set Γ_N of *labeled nulls* (used as placeholders for unknown values, and thus can be also seen as globally existentially quantified variables), and a set Γ_V of regular *variables* (used in dependencies). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a fixed well-ordering of $\Gamma \cup \Gamma_N$ such that every value in Γ_N follows all those in Γ. We denote by $\mathbf{X}$ sequences (or sets, with a slight abuse of notation) of variables $X_1, \ldots, X_k$, with $k \geqslant 0$. Throughout, let $[n] = \{1, \ldots, n\}$, for any integer $n \geqslant 1$.

A *relational schema* $\mathcal{R}$ (or simply *schema*) is a set of *relational symbols* (or *predicates*), each with its associated arity. We write r/n to denote that the predicate r has

[4] This is the complexity class of recognizing words in languages defined by constant-depth Boolean circuits with an (unlimited fan-in) AND and OR gates.

arity n. A *term* t is a constant, null, or variable. An *atomic formula* (or simply *atom*) has the form $r(t_1, \ldots, t_n)$, where r/n is a relation, and $t_1, \ldots, t_n$ are terms. For an atom $\underline{a}$, we denote as $terms(\underline{a})$ and $var(\underline{a})$ the set of its terms and the set of its variables, respectively. These notations naturally extend to sets and conjunctions of atoms. The arity of an atom is the arity of its predicate. An atom is called a *fact* if all of its terms are constants of Γ. Conjunctions and disjunctions of atoms are often identified with the sets of their atoms. An *instance* I for a schema $\mathcal{R}$ is a (possibly infinite) set of atoms of the form $r(\mathbf{t})$, where $r/n \in \mathcal{R}$ and $\mathbf{t} \in (\Gamma \cup \Gamma_N)^n$. A *database* D is a finite instance such that $terms(D) \subset \Gamma$.

A *substitution* from a set of symbols S to a set of symbols S' is a function $h : S \to S'$ defined as follows: $\varnothing$ is a substitution (empty substitution), and if h is a substitution, then $h \cup \{s \to s'\}$ is a substitution, where $s \in S$ and $s' \in S'$. If $s \to s' \in h$, then we write $h(s) = s'$. The *restriction* of h to $T \subseteq S$, denoted $h|_T$, is the substitution $h' = \{t \to h(t) \mid t \in T\}$. A *homomorphism* from a set of atoms A to a set of atoms A' is a substitution $h : \Gamma \cup \Gamma_N \cup \Gamma_V \to \Gamma \cup \Gamma_N \cup \Gamma_V$ such that: if $t \in \Gamma$, then $h(t) = t$, and if $r(t_1, \ldots, t_n) \in A$, then $h(r(t_1, \ldots, t_n)) = r(h(t_1), \ldots, h(t_n)) \in A'$. An *isomorphism* between A and A' is a bijective homomorphism h such that $h(A) = A'$, h^{-1} is a homomorphism, and $h^{-1}(A') = A$.

Disjunctive TGDs. The set of (quantifier-free, positive) AND-OR formulas over a relational schema $\mathcal{R}$ is defined as the set of all (finite) formulas over $\mathcal{R}$ containing $\wedge$ and $\vee$. It is well-known that each AND-OR formula can be converted into an equivalent one in *disjunctive normal form (DNF)*. Given an AND-OR formula ψ, we denote by $\mathit{DNF}(\psi)$ the set of formulas in disjunctive normal form into which ψ can be converted. A *disjunctive tuple-generating dependency (DTGD)* σ over a schema $\mathcal{R}$ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \psi(\mathbf{X}, \mathbf{Z})$, where φ is an AND formula over $\mathcal{R}$, ψ is an AND-OR formula over $\mathcal{R}$, and $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z} \subset \Gamma_V$. Formula φ is the *body* of σ, denoted as $\mathit{body}(\sigma)$, while ψ is the *head* of σ, denoted as $\mathit{head}(\sigma)$. We define the *size* of σ as the sum of the arities of its atoms. In the rest of the paper, for brevity, we will omit universal quantifiers in front of DTGDs, and use a comma for conjoining atoms instead of $\wedge$. Such σ is satisfied by an instance I, written as $I \models \sigma$, if the following holds: whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X}, \mathbf{Y})) \subseteq I$, then there exists $\psi' \in \mathit{DNF}(\psi)$ and a homomorphism $h' \supseteq \{X \to t \mid X \to t \in h|_{\mathbf{X}},$ where $X \in \mathbf{X}$ and $t \in \Gamma \cup \Gamma_N\}$ such that h' maps at least one disjunct of ψ' into I. An instance I satisfies a set Σ of DTGDs, denoted $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$.

Atom Entailment under DTGDs. Given a database D for a schema $\mathcal{R}$, and a set Σ of DTGDs over $\mathcal{R}$, a *model* of D and Σ is an instance I for $\mathcal{R}$ such that $I \supseteq D$ and $I \models \Sigma$; let $\mathit{mods}(D, \Sigma)$ be the set of models of D and Σ. An atom $\underline{a}$ over $\mathcal{R}$ is *entailed* by D and Σ, denoted as $D \cup \Sigma \models \underline{a}$, if for each $M \in \mathit{mods}(D, \Sigma)$, there exists a homomorphism h_M such that $h_M(\underline{a}) \in M$. The central decision problem tackled in this work, called *atom entailment*, is defined as follows:

ATOM-ENTAILMENT

Instance : $\langle \underline{a}, D, \Sigma \rangle$, where $\underline{a}$ is an atom over a schema $\mathcal{R}$, D is a database for $\mathcal{R}$, and Σ is a set of DTGDs over $\mathcal{R}$.

Question : $D \cup \Sigma \models \underline{a}$?

It is well-known that the above problem is undecidable already for *tuple-generating dependencies (TGDs)*, i.e., DTGDs where the head is an AND formula. More precisely, atom entailment is undecidable even under a fixed set of TGDs [12]. The proof of this result hinges on the fact that, with appropriate input atoms, we can simulate the behavior of a deterministic Turing machine using a fixed set of TGDs. Also, as established recently [7], it is possible to encode any set of TGDs using a single TGD.

From the above discussion, we conclude that the recognition of expressive decidable classes of TGDs is a challenging problem. Nowadays, an important research objective is to identify more expressive formalisms under which atom entailment (and other reasoning tasks) is still decidable (see, e.g., [7,14,26]). Known decidable formalisms, which are of special interest for the current work, are the classes of guarded and linear TGDs [13]. A TGD σ is *guarded* if in $body(\sigma)$ there exists an atom $\underline{a}$ such that $var(\underline{a}) = var(body(\sigma))$, i.e., $\underline{a}$ contains all the universally quantified variables of σ. If $body(\sigma)$ contains a single atom, then σ is called *linear*; notice that a linear TGD is trivially guarded since the single body-atom contains all the universally quantified variables. Guarded and linear TGDs can be naturally extended to guarded and linear DTGDs, respectively. In the rest of the paper, we investigate the complexity of atom entailment under guarded and linear DTGDs. Following Vardi's taxonomy [31], the *data complexity* of atom entailment is the complexity calculated taking only the database as input, while the atom and the set of dependencies are considered fixed. The *combined complexity* is the complexity calculated considering as input, together with the database, also the atom and the set of dependencies.

Normalization of DTGDs. In order to simplify the technical definitions and proofs in the rest of the paper, a normal form of DTGDs is adopted. Given a set Σ of DTGDs, we denote by $\mathsf{Normalize}(\Sigma)$ the set of DTGDs obtained by transforming Σ as follows. First, we construct the set Σ' by applying exhaustively on Σ the following rules until each DTGD has in its head either a single atom or a disjunction of two atoms:

1. A DTGD of the form $\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \psi_1(\mathbf{X}_1,\mathbf{Z}_1) \wedge \psi_2(\mathbf{X}_2,\mathbf{Z}_2)$ is replaced by $\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, r^\star(\mathbf{X},\mathbf{Z})$ and $r^\star(\mathbf{X},\mathbf{Z}) \rightarrow \psi_i(\mathbf{X}_i,\mathbf{Z}_i)$, for each $i \in [2]$, where $r^\star$ is an $|\mathbf{X} \cup \mathbf{Z}|$-ary auxiliary predicate not introduced so far.
2. A DTGD of the form $\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \psi_1(\mathbf{X}_1,\mathbf{Z}_1) \vee \psi_2(\mathbf{X}_2,\mathbf{Z}_2)$ is replaced by $\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \vee_{i\in[2]}\, r_i^\star(\mathbf{X}_i,\mathbf{Z}_i)$ and $r_i^\star(\mathbf{X}_i,\mathbf{Z}_i) \rightarrow \psi_i(\mathbf{X}_i,\mathbf{Z}_i)$, for each $i \in [2]$, where $r_i^\star$ is an $|\mathbf{X}_i \cup \mathbf{Z}_i|$-ary auxiliary predicate not introduced so far.

$\mathsf{Normalize}(\Sigma)$ is obtained by transforming Σ' in such a way that eventually each DTGD has at most one existentially quantified variable that occurs only once. In particular, for each $\sigma \in \Sigma'$, assuming that $var(body(\sigma)) \cap var(head(\sigma)) = \mathbf{X}$ and $var(head(\sigma)) \setminus var(body(\sigma)) = \{Z_1,\ldots,Z_n\}$, where $n \geqslant 1$, σ is replaced by

$$\begin{aligned} body(\sigma) &\rightarrow \exists Z_1\, r_1^\star(\mathbf{X},Z_1) \\ r_i^\star(\mathbf{X},Z_1,\ldots,Z_i) &\rightarrow \exists Z_{i+1}\, r_{i+1}^\star(\mathbf{X},Z_1,\ldots,Z_{i+1}), \quad \text{for each } i \in [n-1] \\ r_n^\star(\mathbf{X},Z_1,\ldots,Z_n) &\rightarrow head(\sigma), \end{aligned}$$

where $r_i^\star$ is an $(|\mathbf{X}|+i)$-ary auxiliary predicate, for each $i \in [n]$. It is easy to verify that, during the above construction, we introduce polynomially many auxiliary predicates, and the arity of those predicates is at most the maximum number of variables that appear in the head of a DTGD of Σ. Thus, $\mathsf{Normalize}(\Sigma)$ can be constructed in polynomial time. Finally, it is important to say that the normalized set is (w.r.t. atom entailment) equivalent to the original set. In the rest of the paper, we will state explicitly when we consider normalized sets of DTGDs.

3 Guarded Disjunctive Tuple-Generating Dependencies

The present section weaves together existing results, achieved in the areas of both logics and database theory, to establish the complexity of atom entailment under guarded DTGDs. For the upper bounds, we exploit the *guarded fragment of first-order logic* [3,23], while for the lower bounds we discuss relevant reductions that have already been provided for well-known subclasses of DTGDs.

Interestingly, every set Σ of guarded DTGDs can be translated in logarithmic space into an equivalent guarded first-order theory denoted as $\mathsf{GFO}(\Sigma)$. Given an instance $\langle \underline{a}, D, \Sigma \rangle$ of atom entailment, where Σ is a set guarded DTGDs, it is well-known that $D \cup \Sigma \models \underline{a}$ iff $\mathsf{GFO}(\Sigma) \wedge \psi_D \wedge \neg \underline{a}$ is unsatisfiable, where $\psi_D = \bigwedge_{\underline{d} \in D} \underline{d}$. Since the theory $\mathsf{GFO}(\Sigma) \wedge \psi_D \wedge \neg \underline{a}$ is guarded, we can exploit existing results regarding unsatisfiability of guarded formulas. The complexity of this problem was first investigated by Grädel [23]. In particular, it is in **2EXPTIME** in the general case and in **EXPTIME** in case of fixed arity. The former result was obtained by exhibiting an alternating exponential space algorithm. A deterministic version of this algorithm runs in time $\mathcal{O}(2^{(|\mathcal{R}|+|\psi|)\cdot \omega^{\omega}})$, where $\mathcal{R}$ is the set of predicates in the formula ψ, and ω is the maximum arity over all predicate symbols of $\mathcal{R}$. In the case of fixed arity, the same algorithm clearly runs in exponential time. In case of a fixed theory combined with an input database (data complexity) the same problem is in **NP** [8]. The idea behind this result can be summarized as follows. Consider a fixed guarded first-order formula ψ over a schema $\mathcal{R}$. The algorithm first pre-computes all possible maximal conjunctions of (pairwise non-isomorphic) literals, called *types*, over $\mathcal{R}$ that satisfy ψ. Since ψ is fixed, also the set of types is fixed, as well as the number of steps required to compute it. For example, if $\psi = \forall X(r(X) \rightarrow p(X))$, then its types are: $r(X) \wedge p(X)$, $\neg r(X) \wedge p(X)$, and $\neg r(X) \wedge \neg p(X)$. Next, given an input database ψ_D for $\mathcal{R}$, the algorithm guesses a suitable set B of constants appearing in ψ_D, constructs the set $S = \{\underline{c} \in \psi_D \mid \mathit{terms}(\underline{c}) \subseteq B\}$, and checks in polynomial time whether at least one type of ψ is consistent with S.

Let us now turn our attention to the lower bounds. In the general case, Calì et al. [12] proved that atom entailment is already **2EXPTIME**-hard for guarded TGDs, by simulating an alternating exponential space Turing machine. Moreover, in the same paper, it was also proved that if we consider predicates of fixed arity, the construction can be adapted to simulate an alternating linear space Turing machine. Regarding data complexity, Alviano et al. [2] recently showed that the well-known **coNP**-complete problem of deciding whether a 3-CNF formula is unsatisfiable can be reduced to atom entailment under the fixed set Σ of guarded DTGDs $\{\mathit{clause}(L_1, L_2, L_3, N_1, N_2, N_3) \rightarrow \bigvee_{i \in [3]} \mathit{assign}(L_i, N_i)\} \cup \{\mathit{assign}(L, N), \mathit{assign}(N, L) \rightarrow \mathit{invalid}(L)\}$. Given a propositional formula ψ in 3-CNF, for each clause of the form $l_1 \vee l_2 \vee l_3$ of ψ we add to the database D the fact $\mathit{clause}(l_1, l_2, l_3, \nu(l_1), \nu(l_2), \nu(l_3))$, where $\nu(l) = \neg x$ if $l = x$, and

$\nu(l) = x$ if $l = \neg x$. It can be shown that ψ is unsatisfiable iff $D \cup \Sigma \models \mathit{invalid}(L)$. The previous discussion is summarized in the following result.

Theorem 1. ATOM-ENTAILMENT *under guarded DTGDs is* ***2EXPTIME****-complete in combined complexity,* ***EXPTIME****-complete if the size of the DTGDs is fixed, and* ***coNP****-complete in data complexity.*

It is interesting to see that every set of $\mathcal{ELU}$ axioms [5] can be easily reduced to an equivalent set (w.r.t. atom entailment) of guarded DTGDs. A set of $\mathcal{ELU}$ axioms can be normalized in such a way that only assertions of the form given in Table 2 can appear (see, e.g., [4]). In the same table we give the translation τ of such axioms to guarded DTGDs. Using τ it is not difficult to show that instance checking under $\mathcal{ELU}$ knowledge bases can be reduced in logarithmic space to atom entailment under guarded DTGDs, while preserving the data complexity. It is well-known that instance checking under $\mathcal{ELU}$ ontologies is **coNP**-complete [25]. Guarded DTGDs are expressive enough to capture also the DL obtained by allowing in $\mathcal{ELU}$ role hierarchies and inverse roles.

4 Linear Disjunctive Tuple-Generating Dependencies

In this section, we study the complexity of atom entailment under linear DTGDs. We show that it is **EXPTIME**-complete in combined complexity, **PTIME**-complete if the DTGDs have fixed size, and in $\mathbf{AC}_0$ in data complexity. The first two upper bounds are obtained by giving an alternating algorithm which runs in polynomial space, and in logarithmic space if the DTGDs have fixed size. The $\mathbf{AC}_0$ upper bound is obtained by reducing atom entailment under linear DTGDs into first-order query evaluation. The lower bounds are established by simulating the behavior of an alternating polynomial space (resp., logarithmic space) Turing machine by means of linear DTGDs. It is important to say that the normalization algorithm defined in Section 2 preserves linearity.

4.1 Upper Bounds

Let us first concentrate on the upper bounds. We reduce an instance $\langle \underline{a}, D, \Sigma \rangle$ of atom entailment to the problem of checking the existence of a proof-tree of an atom $\underline{d} \in D$

Table 2. Translation of $\mathcal{ELU}$ to guarded DTGDs; A, B, C are concept names, R is a role name

$\mathcal{ELU}$ Axiom	Guarded DTGD
$A \sqsubseteq B$	$p_A(X) \rightarrow p_B(X)$
$A \sqcap B \sqsubseteq C$	$p_A(X), p_B(X) \rightarrow p_C(X)$
$A \sqsubseteq \exists R.B$	$p_A(X) \rightarrow \exists Y\, p_R(X,Y), p_B(Y)$
$\exists R.A \sqsubseteq B$	$p_R(X,Y), p_A(Y) \rightarrow p_B(X)$
$A \sqsubseteq B \sqcup C$	$p_A(X) \rightarrow p_B(X) \vee p_C(X)$

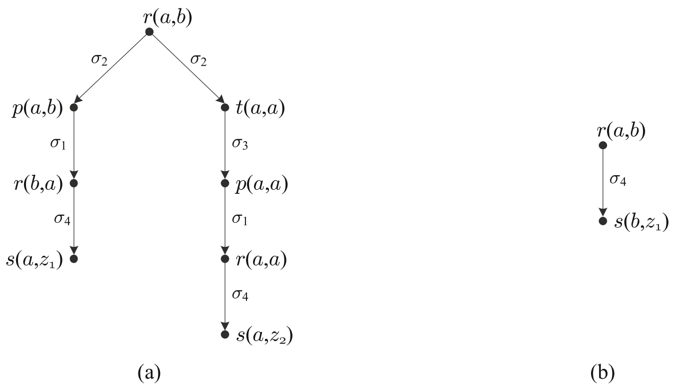

Fig. 1. Possible proof-trees of $r(a, b)$ and Σ

and Σ which is valid w.r.t. $\underline{a}$. Intuitively, such a tree encodes the part of each model of $D \cup \Sigma$ due to which $\underline{a}$ is entailed; the formal definition follows. Consider a fact $\underline{d}$, and a normalized set Σ of linear DTGDs. Let T be a binary tree $\langle N, E, \lambda_1, \lambda_2 \rangle$, where the nodes of N are labeled by λ_1 with atoms that can be formed using predicates occurring in Σ and terms of $terms(\underline{d}) \cup \Gamma_N$, and the edges of E are labeled by $\lambda_2 : E \rightarrow \Sigma$. We say that T is a *proof-tree* of $\underline{d}$ and Σ if:

- The root is labeled by $\underline{d}$.
- For each $v \in N$, there exists $\sigma \in \Sigma$ such that each edge of the form $(u, v) \in E$ is labeled by σ, and the out-degree of v is $|head(\sigma)|$.
- For each $v \in N$, if v has a single outgoing edge (v, u), which is labeled by $p(\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z\, r(\mathbf{X}, Z)$, then there is a homomorphism h such that $h(p(\mathbf{X}, \mathbf{Y})) = \lambda_1(v)$, and there exists $h' = h|_{\mathbf{X}} \cup \{Z \rightarrow t \mid t \in \Gamma_N, t \notin terms(h(p(\mathbf{X}, \mathbf{Y})))\}$ such that $\lambda_1(u) = h'(r(\mathbf{X}, Z))$.
- For each $v \in N$, assuming that the outgoing edges of v are labeled by some σ without an existentially quantified variable, there is a homomorphism h such that $h(body(\sigma)) = \lambda_1(v)$, and $\{h(\underline{b}) \mid \underline{b} \in head(\sigma)\} = \{\lambda_1(u) \mid (v, u) \in E\}$.

A proof-tree T of $\underline{d}$ and Σ is *valid* w.r.t. to an atom $\underline{a}$ with $terms(\underline{a}) \subset \Gamma \cup \Gamma_V$ if, for each leaf u of T, there exists a homomorphism h such that u is labeled by $h(\underline{a})$.

Example 1. Consider the set Σ of linear DTGDs

$$\begin{array}{ll} \sigma_1 : p(X, Y) \rightarrow r(Y, X) & \sigma_2 : r(X, Y) \rightarrow p(X, Y) \vee t(X, X) \\ \sigma_3 : t(X, Y) \rightarrow p(X, Y) & \sigma_4 : r(X, Y) \rightarrow \exists Z\, s(Y, Z). \end{array}$$

Possible proof-trees of $r(a, b)$ and Σ are shown in Figure 1. In particular, tree (a) is valid w.r.t. $s(a, X)$ or $s(X, Y)$, while tree (b) is valid w.r.t. $s(b, X)$ or $s(X, Y)$. □

The following lemma, established in [2], shows that for atom entailment purposes under linear DTGDs, we are allowed to concentrate on a single database atom.

Lemma 1. *Given a database D, a normalized set Σ of linear DTGDs, and an atom $\underline{a}$ with $terms(\underline{a}) \subset \Gamma \cup \Gamma_V$, $D \cup \Sigma \models \underline{a}$ iff there exists $\underline{d} \in D$ such that $\{\underline{d}\} \cup \Sigma \models \underline{a}$.*

Although we can focus on a single database atom, in general we have to consider infinitely many models. The key idea underlying our approach is to use "representatives" of all these models in order to be able to encode them in a single structure, namely, the proof-tree defined above. This can be achieved by considering the skolemized version of the given set of DTGDs. Given a normalized set Σ of linear DTGDs, we define F as the set of *skolem functions* $\{f_\sigma \mid \sigma \in \Sigma\}$, where the arity of each f_σ is the number of universally quantified variables of σ. Let Σ_f denote the set of rules obtained from Σ by replacing each TGD σ of the form $p(\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z\, r(\mathbf{X}, Z)$ with the rule $p(\mathbf{X}, \mathbf{Y}) \rightarrow r(\mathbf{X}, f_\sigma(\mathbf{X}, \mathbf{Y}))$. From well-known results on skolemization the following holds: given a database D, a normalized set Σ of linear DTGDs, and an atom $\underline{a}$ with $terms(\underline{a}) \subset \Gamma \cup \Gamma_V$, $D \cup \Sigma \models \underline{a}$ iff $D \cup \Sigma_f \models \underline{a}$. The set of *skolem terms* Γ_Σ is recursively defined as follows: each term of Γ belongs to Γ_Σ, and if $f_\sigma \in F$ has arity $n > 0$ and $t_1, \ldots, t_n$ are terms of Γ_Σ, then $f_\sigma(t_1, \ldots, t_n) \in \Gamma_\Sigma$. The notion of homomorphism naturally extends to atoms that contain functional terms of the form $f(t_1, \ldots, t_n)$, where each t_i is a variable of Γ_V or a skolem term; in particular, given a homomorphism h, $h(f(t_1, \ldots, t_n)) = f(h(t_1), \ldots, h(t_n))$.

Consider a normalized set Σ of linear DTGDs, an atom $\underline{c}$ with $terms(\underline{c}) \subset \Gamma_\Sigma$, and an instance M of $mods(\{\underline{c}\}, \Sigma_f)$. The *tree* of M is inductively defined as follows: $tree(M)$ is a rooted tree $\langle N, E, \lambda_1, \lambda_2 \rangle$, where N is the set of nodes, E is the edge set, $\lambda_1 : N \rightarrow M$ is a node-labeling function, and $\lambda_2 : E \rightarrow \Sigma_f$ is an edge-labeling function. The root of $tree(M)$ is labeled by $\underline{c}$. Consider a node $v \in N$, and a rule ρ of the form $\underline{b} \rightarrow \underline{b}_1 \vee \underline{b}_2$ or $\underline{b} \rightarrow \underline{b}_1$ such that there is a homomorphism h that maps $\underline{b}$ into $\lambda_1(v)$.

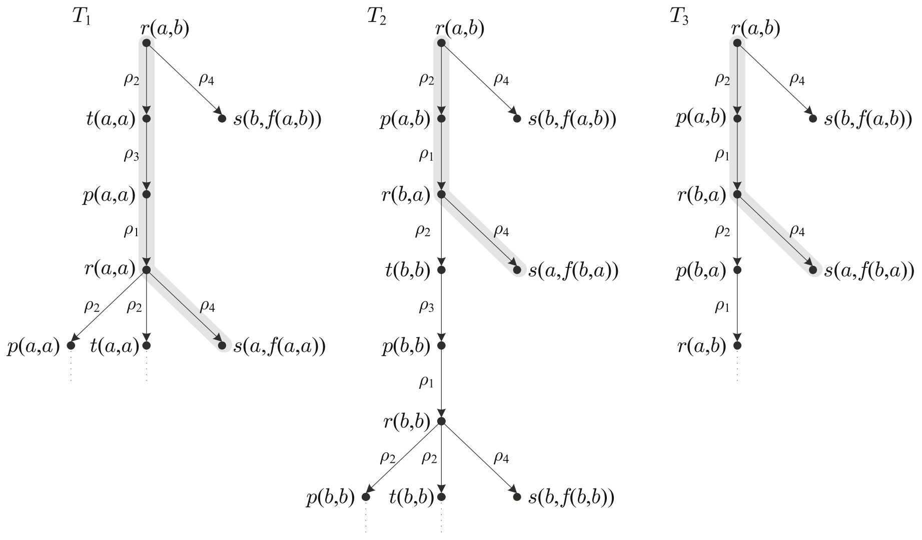

Fig. 2. Possible trees of $trees(\underline{d}, \Sigma_f)$

For each $i \in \{1, 2\}$, if $h(\underline{b}_i) \in M$, then there is a $u \in N$ labeled by $h(\underline{b}_i)$, $e = (v, u)$ belongs to E, and $\lambda_2(e) = \rho$. Notice that, by construction, $\{\lambda_1(v)\}_{v \in N} \subseteq M$. The *$\rho$-tree* of M, denoted as $\rho\text{-}tree(M)$, is obtained from $tree(M)$ by keeping the root node v, each edge $e = (v, u)$ which is labeled by ρ, and the subtree rooted at u. Let $(\rho\text{-})trees(\underline{c}, \Sigma_f) = \{(\rho\text{-})tree(M) \mid M \in mods(\{\underline{c}\}, \Sigma_f)\}$. Notice that, by abuse of notation, given a (ρ-)tree T and an atom $\underline{a}$, we write $T \models \underline{a}$ if $\{\lambda_1(v)\}_{v \in N} \models \underline{a}$. It is not hard to see that $\{\underline{c}\} \cup \Sigma_f \models \underline{a}$ iff $T \models \underline{a}$, for each $T \in trees(\underline{c}, \Sigma_f)$.

Example 2. Consider the set Σ_f of rules

$$\begin{array}{ll} \rho_1 : p(X,Y) \rightarrow r(Y,X) & \rho_2 : r(X,Y) \rightarrow p(X,Y) \vee t(X,X) \\ \rho_3 : t(X,Y) \rightarrow p(X,Y) & \rho_4 : r(X,Y) \rightarrow s(Y, f(X,Y)). \end{array}$$

Possible models of $\{r(a,b)\} \cup \Sigma_f$ are:

$$\begin{array}{l} M_1 = \{r(a,b), s(b, f(a,b)), t(a,a), p(a,a), r(a,a), s(a, f(a,a))\}, \\ M_2 = \{r(a,b), s(b, f(a,b)), p(a,b), r(b,a), t(b,b), p(b,b), r(b,b), s(a, f(b,a)), s(b, f(b,b))\}, \\ M_3 = \{r(a,b), s(b, f(a,b)), p(a,b), r(b,a), p(b,a), s(a, f(b,a))\}. \end{array}$$

Notice that, for each other model M of $\{r(a,b)\} \cup \Sigma_f$, it holds that $M_i \subseteq M$, for at least one $i \in [3]$; the tree T_i of M_i is depicted in Figure 2. Observe that each T_i has exactly one ρ_2-tree and one ρ_4-tree. Moreover, the shaded paths form (modulo null renaming) the proof-tree of $r(a,b)$ and Σ_f shown in Figure 1(a), which is valid w.r.t. $s(a, X)$. □

Observe that the first edge of each shaded path in Figure 2 is labeled by the same rule. As shown in the next technical lemma, this is a general property that holds whenever the given database and set of DTGDs entail the given atom.

Lemma 2. *Consider a normalized set Σ of linear DTGDs, an atom $\underline{c}$ with $terms(\underline{c}) \subset \Gamma_\Sigma$, and an atom $\underline{a}$ with $terms(\underline{a}) \subset \Gamma \cup \Gamma_V$. It holds that $\{c\} \cup \Sigma_f \models \underline{a}$ iff there exists $\rho \in \Sigma_f$ such that $T \models \underline{a}$, for each $T \in \rho\text{-}trees(\underline{c}, \Sigma_f)$.*

Proof (sketch). By definition of the tree of a model, $trees(\underline{c}, \Sigma_f)$ is isomorphic to the set $\mathcal{T} = \{\uplus_{i \in [|\Sigma|]} T_i \mid \langle T_1, \ldots, T_{|\Sigma|}\rangle \in \times_{\rho \in \Sigma_f} \rho\text{-}trees(\underline{c}, \Sigma_f)\}$, where $\uplus_{i \in [|\Sigma|]} T_i$ is the rooted tree obtained from the disjoint union of $T_1, \ldots, T_{|\Sigma|}$ after merging the root nodes (see Figure 3). We are now ready to show our lemma. ($\Rightarrow$) Assume that for each $\rho \in \Sigma_f$ there exists $T \in \rho\text{-}trees(\underline{c}, \Sigma_f)$ such that $T \not\models \underline{a}$. Therefore, there exists $T' \in \mathcal{T}$ such that $T' \not\models \underline{a}$. Clearly, there is $T'' \in trees(\underline{c}, \Sigma_f)$ isomorphic to T'. Assuming that $T'' = tree(M)$, for some $M \in mods(\{\underline{c}\}, \Sigma_f)$, $M \not\models \underline{a}$; thus, $\{\underline{c}\} \cup \Sigma_f \not\models \underline{a}$. ($\Leftarrow$) This direction follows immediately. □

Given a fact $\underline{d}$, a set Σ of linear DTGDs, and an atom $\underline{a}$, by applying recursively the property provided by Lemma 2, one can build a proof-tree of $\underline{d}$ and Σ which is valid w.r.t. $\underline{a}$. This is exactly the key idea underlying the proof of the next result.

Lemma 3. *Consider a database D, a normalized set Σ of linear DTGDs, and an atom $\underline{a}$ with $terms(\underline{a}) \subset \Gamma \cup \Gamma_V$. It holds that $D \cup \Sigma \models \underline{a}$ iff there exists a proof-tree of $\underline{d}$ and Σ, for some $\underline{d} \in D$, which is valid w.r.t. $\underline{a}$.*

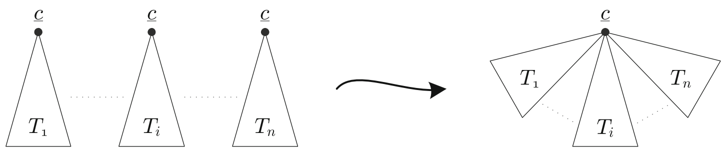

Fig. 3. Rooted tree obtained from the disjoint union of $T_1, \ldots, T_n$ after merging the root nodes

Proof (sketch). ($\Rightarrow$) The claim is shown by giving a proof-tree of some $\underline{d} \in D$ and Σ which is valid w.r.t. $\underline{a}$. By Lemmas 1 and 2, there are $\underline{d} \in D$ and $\rho \in \Sigma_f$ such that in every $T \in \rho\text{-}trees(\underline{d}, \Sigma_f)$ there is a path π_T from the root to a node u which is labeled by $h(\underline{a})$, where h is a homomorphism, and the first edge e_T of π_T is labeled by a rule ρ of the form $\underline{b} \to \underline{b}_1 \vee \underline{b}_2$ or $\underline{b} \to \underline{b}_1$. Let us now construct a rooted binary tree $T' = \langle N, E, \lambda_1, \lambda_2 \rangle$; initially, $N = \{v\}$, $E = \varnothing$, $\lambda_1 = \{v \to \underline{d}\}$, and $\lambda_2 = \varnothing$. Clearly, $\rho\text{-}trees(\underline{d}, \Sigma_f)$ can be partitioned into S_1 and S_2 such that, for each $T \in S_1$, if $e_T = (u, w)$ and h maps $\underline{b}$ into $\underline{d}$ (which is the label of u), then w is labeled by $h(\underline{b}_1)$. For each $i \in \{1, 2\}$, assume that the second node of π_T, for each $T \in S_i$, is labeled by $\underline{g}_i$. Let $N = N \cup \{w_1, w_2\}$, $E = E \cup \{(v, w_i)\}_{i \in \{1,2\}}$, $\lambda_1 = \lambda_1 \cup \{w_i \to \underline{g}_i\}_{i \in \{1,2\}}$, and $\lambda_2 = \lambda_2 \cup \{(v, w_i) \to \rho\}_{i \in \{1,2\}}$. Due to the fact that $\{\underline{g}_i\} \cup \Sigma_f \models \underline{a}$, for each $i \in [2]$, Lemma 2 can be recursively applied finitely many times as described above, and eventually T' is constructed. The desired proof-tree is obtained from T' by replacing the skolem terms occurring in T' with distinct nulls of Γ_N.

($\Leftarrow$) By hypothesis, there exists $\underline{d} \in D$ and a proof-tree T of $\underline{d}$ and Σ which is valid w.r.t. $\underline{a}$. It is possible to construct from T a rooted binary tree T' such that the nodes are labeled by atoms with skolem terms (instead of nulls), the edges are labeled by rules of Σ_f, and the structural properties of T are preserved. Assume that the label of each outgoing edge of the root of T' is ρ. By Lemma 1 and 2, it suffices to show that each ρ-tree of $\underline{d}$ and Σ_f entails $\underline{a}$. This follows from the fact that, for each $T'' \in \rho\text{-}trees(\underline{d}, \Sigma_f)$, there is a path from the root to a leaf of T' that can be mapped into T''. □

Interestingly, a fact $\underline{d}$ and a normalized set Σ of linear DTGDs admit a valid proof-tree w.r.t. an atom $\underline{a}$ iff they admit a "small" valid proof-tree T w.r.t. $\underline{a}$ involving at most $\omega + 1$ nulls, where ω is the maximum arity over all predicates of Σ. This holds since, by definition, T can be constructed in such a way that any pair of edges of the form (v, u) and (v, u') involves at most $\omega + 1$ nulls. By combining this observation with Lemma 3, we get that atom entailment is equivalent to the problem of deciding whether a "small" proof-tree exists. We do this by applying the alternating algorithm SearchPT given in Figure 4; an example of the computation of the algorithm follows.

Example 3. Consider the instance $I = \langle \underline{a}, \{r(a, b)\}, \Sigma \rangle$ of atom entailment, where Σ is:

$$\begin{array}{ll} \sigma_1 : r(X,Y) \to p(X) \vee t(Y,X) & \sigma_2 : p(X) \to r(X,X) \\ \sigma_3 : t(X,Y) \to \exists Z\, s(Y,Z) & \sigma_4 : s(X,Y) \to \exists Z\, s(Y,Z). \end{array}$$

Figure 5 shows an initial part of the alternating computation of SearchPT on I. Observe that in the shaded edge exactly three (maximum arity plus one) nulls appear. □

```
Algorithm SearchPT($\underline{a}, D, \Sigma$)

Input: An instance $\langle \underline{a}, D, \Sigma \rangle$ of ATOM-ENTAILMENT

1. $N := \{z_i \in \Gamma_N\}_{i \in [\omega+1]}$, and let $\rho : var(\underline{a}) \rightarrow N$ be a one-to-one substitution;
2. Existentially choose $\underline{c} \in D$;
3. Existentially choose to either execute step 4 or to skip to step 5;
4. If there exists a homomorphism $h$ such that $h(\underline{a}) = \underline{c}$, then accept;
5. Existentially choose $\sigma \in \Sigma$ and a homomorphism $h$ such that $h(body(\sigma)) = \underline{c}$; if there is
   no such a pair, then reject;
6. If $\sigma = p(\mathbf{X}, \mathbf{Y}) \rightarrow r_1(\mathbf{X}_1) \vee r_2(\mathbf{X}_2)$, then $\underline{b}_1 := h(r_1(\mathbf{X}_1))$ and $\underline{b}_2 := h(r_2(\mathbf{X}_2))$;
7. If $\sigma = p(\mathbf{X}, \mathbf{Y}) \rightarrow r(\mathbf{X})$, then $\underline{b}_1 := h(r(\mathbf{X}, Z))$ and $\underline{b}_2 := \rho(\underline{a})$;
8. If $\sigma = p(\mathbf{X}, \mathbf{Y}) \rightarrow \exists Z\, r(\mathbf{X}, Z)$, then $\underline{b}_1 := h'(r(\mathbf{X}, Z))$, where $h'$ is the homomorphism
   $h|_{\mathbf{X}} \cup \{Z \rightarrow t \mid t \in N,\ t \notin terms(h(p(\mathbf{X}, \mathbf{Y})))\}$, and $\underline{b}_2 := \rho(\underline{a})$;
9. Universally choose $\underline{c} \in \{\underline{b}_1, \underline{b}_2\}$ and goto step 3.
```

Fig. 4. The alternating algorithm SearchPT

Soundness and completeness of SearchPT follow by construction.

Proposition 1. *Given an instance $\langle \underline{a}, D, \Sigma \rangle$ of* ATOM-ENTAILMENT*, where Σ is a set of linear DTGDs, $D \cup \Sigma \models \underline{a}$ iff* SearchPT($\underline{a}, D$, Normalize(Σ)) *accepts.*

Equipped with the above machinery, we are now ready to establish the desired complexity upper bounds of our problem.

Theorem 2. ATOM-ENTAILMENT *under linear DTGDs is in **EXPTIME** in combined complexity, and in **PTIME** if the DTGDs have fixed size.*

Proof. Consider an instance $\langle \underline{a}, D, \Sigma \rangle$ of ATOM-ENTAILMENT, where Σ is a set of linear DTGDs. Since SearchPT is an alternating algorithm and $\Sigma' =$ Normalize(Σ) can be computed in polynomial time, to obtain the desired upper bounds, by Proposition 1, it suffices to show that SearchPT($\underline{a}, D, \Sigma'$) runs in polynomial space in the general case, and in logarithmic space in the restricted case. At each step of the computation we need to remember at most two atoms involving ω terms, where each term can be represented using logarithmically many bits; more precisely, we need $\mathcal{O}(\omega \log \omega + \omega \log n)$ space, where $n = |terms(D)|$. Notice that, if the DTGDs of Σ have fixed size, then ω is also fixed (see Section 2), and the claim follows. □

The rest of this subsection is devoted to show that atom entailment under linear DTGDs is in $\mathbf{AC}_0$ in data complexity. We do this by establishing that linear DTGDs are first-order rewritable, i.e., atom entailment under linear DTGDs can be reduced to the problem of evaluating a first-order query over a database. First-order rewritability was first introduced in the context of description logics [18].

Consider a normalized set Σ of linear DTGDs over a schema $\mathcal{R}$, and an atom $\underline{a}$ over $\mathcal{R}$. Let C be the constants occurring in $\underline{a}$, and $N = \{z_1, \ldots, z_\omega\}$ be a set of nulls, where ω is the maximum arity over all predicates of $\mathcal{R}$. We call $base(\underline{a}, \mathcal{R})$ the

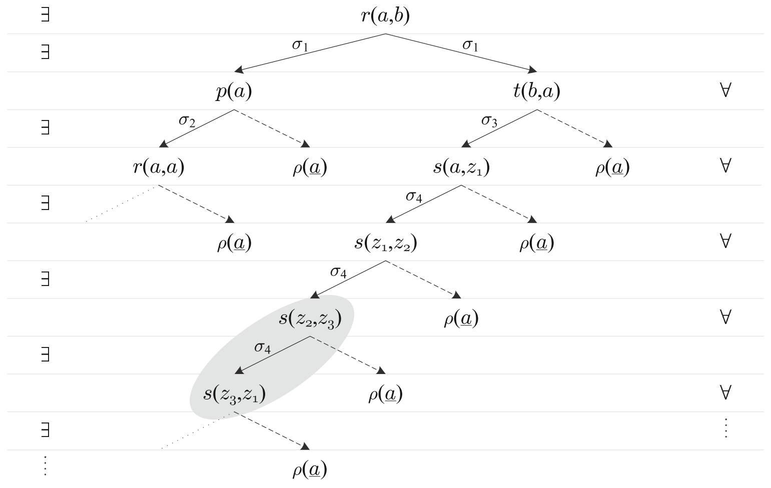

Fig. 5. Computation of the algorithm SearchPT

set of all atoms that can be formed with predicates of $\mathcal{R}$ and terms of $C \cup N$. Let $B = \{\underline{b} \mid \underline{b} \in \mathit{base}(\underline{a}, \mathcal{R}),\ \{\underline{b}\} \cup \Sigma \models \underline{a}\}$, and ρ be a renaming substitution that maps each $z \in N$ into a distinct variable $X_z \in \Gamma_V$. Let $Q[\underline{a}, \Sigma]$ be the first-order query $\exists X_{z_1} \ldots \exists X_{z_\omega} \bigvee_{\underline{b} \in B} \rho(\underline{b})$. The size of $Q[\underline{a}, \Sigma]$, defined as the number of disjuncts, is at most $|\mathcal{R}| \cdot (2\omega)^\omega$. Since, by Theorem 2, atom entailment under linear DTGDs is feasible in exponential time, the rewriting can be also constructed in exponential time. Notice that a database D entails $Q[\underline{a}, \Sigma]$, denoted $D \models Q[\underline{a}, \Sigma]$, if there exists a homomorphism h that maps $\rho(\underline{b})$ into D, for at least one atom $\underline{b} \in B$. In what follows, we show that $Q[\underline{a}, \Sigma]$ is a sound and complete rewriting of $\underline{a}$ and Σ.

Lemma 4. *Consider a database D, a normalized set Σ of linear DTGDs, and an atom $\underline{a}$ with $\mathit{terms}(\underline{a}) \subset \Gamma \cup \Gamma_V$. It holds that $D \cup \Sigma \models \underline{a}$ iff $D \models Q[\underline{a}, \Sigma]$.*

Proof (sketch). By construction of $Q[\underline{a}, \Sigma]$, it suffices to show that $D \cup \Sigma \models \underline{a}$ iff there exists $\underline{b} \in \mathit{base}(\underline{a}, \mathcal{R})$ and a homomorphism h such that $h(\underline{b}) \in D$ and $\{\underline{b}\} \cup \Sigma \models \underline{a}$. ($\Rightarrow$) By Lemma 1, there exists $\underline{d} \in D$ such that $\{\underline{d}\} \cup \Sigma \models \underline{a}$; therefore, there exists a homomorphism h_M that maps each $M \in \mathit{mods}(\{\underline{d}\}, \Sigma)$ into $\underline{a}$. Moreover, by construction, there exists $\underline{b} \in \mathit{base}(\underline{a}, \mathcal{R})$ and a bijective homomorphism μ such that $\mu(\underline{b}) = \underline{d}$; clearly, $\mu(\underline{b}) \in D$. Let $f : \mathit{terms}(\underline{d}) \to \mathit{terms}(\underline{b})$ be the substitution $\{t \to t' \mid t' \to t \in \mu|_{\mathit{terms}(\underline{d})}\}$. By induction on the depth of the trees of the models, it can be shown that for each $M' \in \mathit{mods}(\{\underline{b}\}, \Sigma)$, there exists $M \in \mathit{mods}(\{\underline{d}\}, \Sigma)$ such that $f(M) = M'$; thus, the homomorphism $f \circ h_M$ maps $\underline{a}$ into M'. ($\Leftarrow$) This direction can be easily shown by providing a similar argument. $\square$

Since evaluation of first-order queries is feasible in $\mathbf{AC}_0$ [32], Lemma 4 implies the following complexity result.

Theorem 3. ATOM-ENTAILMENT *under linear DTGDs is in $\mathbf{AC}_0$ in data complexity.*

4.2 Lower Bounds

We proceed now to establish the desired complexity lower bounds of atom entailment under linear DTGDs.

Theorem 4. *Consider an atom $\underline{a}$ over a schema $\mathcal{R}$, a database D for $\mathcal{R}$, and a set Σ of linear DTGDs over $\mathcal{R}$. The problem of deciding whether $D \cup \Sigma \models \underline{a}$ is* **EXPTIME***-hard in combined complexity even if $|D| = 1$ and $|\mathcal{R}| = 2$.*

Proof (sketch). To prove our claim, it suffices to simulate the behavior of an **APSPACE** Turing machine M by means of linear DTGDs. W.l.o.g., we assume that M has exactly one accepting state s_{acc}, it is well-behaved and never "falls off" the left end of the tape, and also each configuration of M has at most two subsequent configurations. Moreover, we assume that the tape alphabet of M is $\{0, 1, \sqcup\}$, where $\sqcup$ denotes the blank symbol. Suppose that M halts on input $I = a_1 \ldots a_{|I|}$ using $n = |I|^k$ cells, where $k > 0$. Assume also that $s_1 < \ldots < s_m$, where $m > 0$, is the order that the states appear in the encoding of M, and that s_{init} is the initial state of M. We use a $(2n + m + 6)$-ary predicate called *config* to represent a configuration of M on I. An atom of the form $\mathit{config}(s, c_1, \ldots, c_n, b_1, \ldots, b_n, 0, 1, 0, 1, \sqcup, s_1, \ldots, s_m)$, where $s \in \{s_1, \ldots, s_m\}$, $\langle c_1, \ldots, c_n\rangle \in \{0, 1, \sqcup\}^n$, $\langle b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n\rangle = \{0\}^{n-1}$ and $b_i = 1$, represents the fact that M is in state s, the tape contains the string $c_1, \ldots, c_n$, and the cursor points at the i-th cell. The tuple $\langle 0, 1, 0, 1, \sqcup, s_1, \ldots, s_m\rangle$ (formally not part of the represented configuration) keeps the binary values 0 and 1, which will be used later in the simulation to move the cursor, the tape alphabet, and the states of the machine. Let D be the database that contains the atom

$$\mathit{config}(s_{init}, a_1, \ldots, a_{|I|}, \underbrace{\sqcup, \ldots, \sqcup}_{n-|I|}, 1, \underbrace{0, \ldots, 0}_{n-1}, 0, 1, 0, 1, \sqcup, s_1, \ldots, s_m)$$

which represents the initial configuration of M on I. We now encode the behavior of M on all transition rules that move the cursor to the right (resp., left) in the first (resp., second) subsequent configuration. For notational convenience, let

$$\mathbf{B}_\ell = \underbrace{Z_0, \ldots, Z_0}_{\ell-1}, Z_1, \underbrace{Z_0, \ldots, Z_0}_{n-\ell} \quad \text{and} \quad \mathbf{T} = Z_0, Z_1, V_0, V_1, V_\sqcup, S_1, \ldots, S_m,$$

where $\{Z_0, Z_1, V_0, V_1, V_\sqcup, S_1, \ldots, S_m\} \subset \Gamma_V$. For each transition rule of the form $\langle s_i, a\rangle \rightarrow \langle\langle s_j, b, \rightarrow\rangle, \langle s_k, c, \leftarrow\rangle\rangle$ in the transition function of M, we add in Σ the following linear DTGDs: for each $\ell \in [n-1]$, $\underline{a} \rightarrow \underline{b} \wedge \underline{c}$ if s_i is an $\exists$-state, and $\underline{a} \rightarrow \underline{b} \vee \underline{c}$ if s_i is a $\forall$-state, where

$$\begin{aligned}
\underline{a} &= \mathit{config}(S_i, X_1, \ldots, X_{\ell-1}, V_a, X_{\ell+1}, \ldots, X_n, \mathbf{B}_\ell, \mathbf{T}),\\
\underline{b} &= \mathit{config}(S_j, X_1, \ldots, X_{\ell-1}, V_b, X_{\ell+1}, \ldots, X_n, \mathbf{B}_{\ell+1}, \mathbf{T}),\\
\underline{c} &= \mathit{config}(S_k, X_1, \ldots, X_{\ell-1}, V_c, X_{\ell+1}, \ldots, X_n, \mathbf{B}_{\ell-1}, \mathbf{T}).
\end{aligned}$$

Similar DTGDs are used to encode the transition rules that move the cursor to the left or leave the cursor unmoved. Finally, assuming that $s_{acc} = s_i$, we add in Σ the following DTGDs: for each $\ell \in [n-1]$, $\mathit{config}(S_i, X_1, \ldots, X_n, \mathbf{B}_\ell, \mathbf{T}) \rightarrow \mathit{accept}(S_i)$. Notice that the above construction is feasible in polynomial time. It is not difficult to verify that, if $\underline{a} = accept(A)$, where $A \in \Gamma_V$, then M accepts I iff $D \cup \Sigma \models \underline{a}$. Since **APSPACE** coincides with **EXPTIME**, the claim follows from the fact that all the DTGDs of Σ are linear, the database D contains a single atom, and $\mathcal{R}$ contains only two predicates, namely, *config* and *accept*. $\square$

It is interesting to say that in the above construction the arity of the predicate *config* can be reduced to $2n + 1$ if we allow the use of constants in the DTGDs. If we consider predicates of fixed arity, then the encoding given in the proof of Theorem 4 can be adapted in order to simulate an alternating logarithmic space Turing machine.

Theorem 5. *Consider an atom $\underline{a}$ over a schema $\mathcal{R}$, a database D for $\mathcal{R}$, and a set Σ of linear DTGDs over $\mathcal{R}$. Assuming that each DTGD of Σ has fixed size, the problem of deciding whether $D \cup \Sigma \models \underline{a}$ is **PTIME**-hard. The same lower bound holds even if $|D| = 1$ and each predicate of $\mathcal{R}$ is unary.*

Proof (sketch). It is well-known that **ALOGSPACE** equals **PTIME**. Thus, to prove our claim, it suffices to simulate the behavior of an **ALOGSPACE** Turing machine M on an input I by means of linear DTGDs with predicates of fixed arity. Recall that a logarithmic space Turing machine is equipped with a read-only input tape and a read/write work tape. Assume that M halts on $I = a_1, \ldots, a_{|I|}$ using $n = \log |I|^k$ cells, where $k > 0$. The key idea of the proof is to modify the construction given in the proof of Theorem 4 in such a way that a configuration of M on I is represented as a unary predicate. For example, the database atom can be encoded as

$$\mathit{config}[s_{init}\, \overbrace{a_1..a_{|I|} 1 \underbrace{0..0}_{|I|-1}}^{\text{input tape}} \overbrace{\underbrace{\sqcup..\sqcup}_{n} 1 \underbrace{0..0}_{n-1}}^{\text{work tape}}](c)$$

which represents the initial configuration of M on I, where c is an arbitrary constant of Γ. Since the number of configurations of M on I is polynomial, we need polynomially many unary predicates. The rest of the construction, which is feasible in logarithmic space, can be done by adapting the DTGDs given in the proof of Theorem 4. $\square$

Acknowledgements. This research has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement DIADEM no. 246858, and from the Oxford Martin School's grant no. LC0910-019. Marco Manna's work was supported by the European Commission through the European Social Fund and by Calabria Region.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Alviano, M., Faber, W., Leone, N., Manna, M.: Disjunctive Datalog with existential quantifiers: Semantics, decidability, and complexity issues. In: TPLP (to appear, 2012)

3. Andréka, H., Németi, I., van Benthem, J.: Modal languages and bounded fragments of predicate logic. J. Philosophical Logic 27(3), 217–274 (1998)
4. Baader, F.: Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In: Proc. of IJCAI, pp. 319–324 (2003)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of IJCAI, pp. 364–369 (2005)
6. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
7. 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)
8. Bárány, V., Gottlob, G., Otto, M.: Querying the guarded fragment. In: Proc. of LICS, pp. 1–10 (2010)
9. Beeri, C., Vardi, M.Y.: The Implication Problem for Data Dependencies. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 73–85. Springer, Heidelberg (1981)
10. Berardi, D., Calvanese, D., Giacomo, G.D.: Reasoning on UML class diagrams. Artif. Intell. 168(1-2), 70–118 (2005)
11. Cabibbo, L.: The expressive power of stratified logic programs with value invention. Inf. Comput. 147(1), 22–56 (1998)
12. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR, pp. 70–80 (2008)
13. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. In: Proc. of PODS, pp. 77–86 (2009)
14. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog$^{\pm}$: A family of logical knowledge representation and query languages for new applications. In: Proc. of LICS, pp. 228–242 (2010)
15. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. VLDB 3(1), 554–565 (2010)
16. Calì, A., Gottlob, G., Pieris, A.: Query Answering under Non-guarded Rules in Datalog$^{\pm}$. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)
17. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of KR, pp. 260–270 (2006)
18. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3) (2007)
19. Chen, P.P.: The Entity-Relationship model - Toward a unified view of data. ACM Trans. Database Syst. 1(1), 9–36 (1976)
20. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proc. of PODS, pp. 149–158 (2008)
21. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Trans. Database Syst. 22(3), 364–418 (1997)
22. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
23. Grädel, E.: On the restraining power of guards. J. Symb. Log. 64(4), 1719–1742 (1999)
24. Johnson, D.S., Klug, A.C.: Testing containment of conjunctive queries under functional and inclusion dependencies. J. Comput. Syst. Sci. 28(1), 167–189 (1984)
25. Krisnadhi, A., Lutz, C.: Data complexity in the EL family of DLs. In: Proc. of DL (2007)
26. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proc. of IJCAI, pp. 963–968 (2011)

27. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: Proc. of KR, pp. 13–23 (2012)
28. Marnette, B.: Generalized schema-mappings: From termination to tractability. In: Proc. of PODS, pp. 13–22 (2009)
29. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the semantic web. J. Web Sem. 5(4), 240–250 (2007)
30. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
31. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Proc. of STOCS, pp. 137–146 (1982)
32. Vardi, M.Y.: On the complexity of bounded-variable queries. In: Proc. of PODS, pp. 266–276 (1995)