A General Datalog-Based Framework for Tractable Query Answering over Ontologies[☆]

Andrea Calì[a,b], Georg Gottlob[c,b], Thomas Lukasiewicz[c]

[a] *Department of Information Systems and Computing, Brunel University, UK*
[b] *Oxford-Man Institute of Quantitative Finance, University of Oxford, UK*
[c] *Computing Laboratory, University of Oxford, UK*

Abstract

Ontologies and rules play a central role in the development of the Semantic Web. Recent research in this context focuses especially on highly scalable formalisms for the Web of Data, which may highly benefit from exploiting database technologies. In this paper, as a first step towards closing the gap between the Semantic Web and databases, we introduce a family of expressive extensions of Datalog, called Datalog$^{\pm}$, as a new paradigm for query answering over ontologies. The Datalog$^{\pm}$ family admits existentially quantified variables in rule heads, and has suitable restrictions to ensure highly efficient ontology querying. We show in particular that different versions of Datalog$^{\pm}$ generalize the tractable description logic $\mathcal{EL}$ and the *DL-Lite* family of tractable description logics, which are the most common tractable ontology languages in the context of the Semantic Web and databases. We also show how stratified negation can be added to Datalog$^{\pm}$ while keeping ontology querying tractable. Furthermore, the Datalog$^{\pm}$ family is of interest in its own right, and can, moreover, be used in various contexts such as data integration and data exchange. It paves the way for applying results from databases to the context of the Semantic Web.

Keywords: Datalog, ontologies, Semantic Web, conjunctive queries, query evaluation, chase, dependencies, constraints, complexity, tractability.

[☆] This paper is a significantly extended and revised version of a paper that appeared in: *Proceedings of the 28th ACM Symposium on Principles of Database Systems (PODS 2009)*, pp. 77–86, ACM Press, 2009 [21].

Email addresses: andrea.cali@oxford-man.ox.ac.uk (Andrea Calì), georg.gottlob@comlab.ox.ac.uk (Georg Gottlob), thomas.lukasiewicz@comlab.ox.ac.uk (Thomas Lukasiewicz)

1. Introduction

Ontology languages, rule-based systems, and their integrations play a central role in the development of the Semantic Web [15]. Although there are a plethora of approaches to tight and loose (or hybrid) integrations of ontology languages and rule-based systems, and to generalizations of ontology languages by the ability to express rules, there is literally no previous work on how to generalize database rules and dependencies so that they can express ontological axioms. This is surprising, especially also because there are recently strong interests in the Semantic Web community on highly scalable formalisms for the *Web of Data*, which would benefit very much from applying technologies and results from databases.

In this paper, we try to fill this gap. We propose and study variants of Datalog that are suited for efficient ontological reasoning, and, in particular, for tractable ontology-based query answering. We introduce the Datalog$^{\pm}$ family of Datalog variants, which extend plain Datalog by the possibility of existential quantification in rule heads, and by a number of other features, and, at the same time, restrict the rule syntax in order to achieve tractability. The goal of this paper is threefold:

- First, we aim at bridging an apparent gap in expressive power between database query languages and description logics (DLs) as ontology languages, extending the well-known Datalog language in order to embed DLs.
- Second, we aim at transferring important concepts and proof techniques from database theory to DLs. For example, it was so far not clear how to enrich tractable DLs by the feature of nonmonotonic negation. By the results of the present paper, we are now able to enrich DLs by stratified negation via mappings from DLs to Datalog$^{\pm}$ with stratified negation.
- Last but not least, we have a genuine interest in studying new fascinating tractable query languages. We are convinced that these languages are of independent relevance and interest, even without reference to ontological reasoning. Moreover, we have reasons to believe that the languages that we discuss may be useful for data exchange [45], and constraint satisfaction for automatic configuration, where value invention techniques are used [46, 66]. For lack of space, we do not discuss these applications in detail here.

In addition to playing a key role in the development of the Semantic Web, ontologies are also becoming more and more important in the database area, for

example, in data modeling and information integration [58]. While much of the research on DLs of the last decade was centered around decidability issues, there is a current trend towards highly scalable procedures for query answering over ontologies. A family of well-known DLs fulfilling these criteria is, e.g., the *DL-Lite* family [31, 72] (which has recently been further extended in [7]). The following example briefly illustrates how queries can be posed and answered in *DL-Lite*.

Example 1. A DL knowledge base consists of a TBox and an ABox. For example, the knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed by the axioms $\mathit{ConferencePaper} \sqsubseteq \mathit{Article}$ and $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthorOf}$ in the TBox, respectively, while the knowledge that John is a scientist can be expressed by the axiom $\mathit{Scientist}(\mathit{john})$ in the ABox. A simple Boolean conjunctive query (BCQ) asking whether John authors a paper is $\exists X\ \mathit{isAuthorOf}(\mathit{john}, X)$.

An ABox can be identified with an extensional database, while a TBox can be regarded as a set of integrity constraints involving, among others, functional dependencies and (possibly recursive) inclusion dependencies [44, 1]. An important result of [31, 72] is that the *DL-Lite* description logics, in particular, $\mathit{DL\text{-}Lite}_{\mathcal{F}}$, $\mathit{DL\text{-}Lite}_{\mathcal{R}}$, and $\mathit{DL\text{-}Lite}_{\mathcal{A}}$, are not only decidable, but that answering (unions of) conjunctive queries for them is in LOGSPACE, and actually in AC_0, in the data complexity, and query answering in *DL-Lite* is FO-rewritable (see below) [31].

In the context of DLs, *data complexity* is the complexity of query answering over input ABoxes, when both the TBox and the query are fixed. This scenario is very similar to query answering with well-known rule-based languages, such as Datalog. It is easy to see that plain Datalog can neither directly express *DL-Lite* disjointness constraints (e.g., $\mathit{ConferencePaper} \sqsubseteq \neg \mathit{JournalPaper}$), nor the functional constraints used in $\mathit{DL\text{-}Lite}_{\mathcal{F}}$ (e.g., $(\mathsf{funct}\ \mathit{hasFirstAuthor})$). Moreover, as observed in [70], the lack of value creation makes plain Datalog not very well suited for ontological reasoning with inclusion axioms either (e.g., $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthorOf}$). It is thus natural to ask whether Datalog can be suitably modified to nicely accommodate ontological axioms and constraints such as those expressible in the *DL-Lite* family. In particular, we have addressed the following two questions:

Question 1: What are the main modifications of Datalog that are required for ontological knowledge representation and query-answering?

Question 2: Are there versions of Datalog that encompass the *DL-Lite* family of description logics, and that share the favorable data complexity bounds for query-answering with *DL-Lite*? If so, how do they look like?

As an answer to Question 1, we identified the possibility of having *existentially quantified variables* in rule heads as the main Datalog extension enabling ontological knowledge representation and reasoning. Datalog rules extended this way are known as *tuple generating dependencies (TGDs)*, see [14]. Given that fact inference (let alone conjunctive query answering) under TGDs is undecidable [13, 2], we must somehow restrict the rule syntax for achieving decidability. We thus require that the rule bodies of TGDs are *guarded*. This means that in each rule body of a TGD there must exist an atom, called *guard*, in which all non-existentially quantified variables of the rule occur as arguments. An example of a guarded TGD is $P(X) \wedge R(X,Y) \wedge Q(Y) \rightarrow \exists Z\, R(Y,Z)$.

Guarded TGDs form the first Datalog$^{\pm}$ formalism that we consider. Note that this formalization was briefly mentioned in [20]. We embark in Section 3 in a detailed analysis of the data complexity of this formalism. To this aim, we study the behavior of the (oblivious) chase algorithm [65, 14], a well-known algorithm for constructing a (usually infinite) universal model $chase(D, \Sigma)$ of a given extensional database D and a set of guarded TGDs Σ.

As a key lemma, we prove that for each set of guarded TGDs Σ, there exists a constant k such that for every extensional database D and every atom $\mathbf{a}$ generated at some depth level d while chasing D with Σ, such that whenever the same chase generates an atom $\mathbf{b}$ whose arguments are among those of $\mathbf{a}$, then $\mathbf{b}$ must be generated at depth level at most $d + k$. Using this lemma, we can show that whenever a Boolean conjunctive query (BCQ) Q maps homomorphically into $chase(D, \Sigma)$ then it maps into the initial fragment of constant depth $k \times |Q|$ of $chase(D, \Sigma)$. This result is a nontrivial generalization of a classical result by Johnson and Klug [50] on inclusion dependencies, which are a restricted class of guarded TGDs. For the complexity of fact inference and answering BCQs, we then get the following result:

Theorem: Given a database D and a fixed set of guarded TGDs Σ, deciding whether $D \cup \Sigma \models \mathbf{a}$ for facts $\mathbf{a}$ is PTIME-complete and can be done in linear time. Moreover, deciding whether $D \cup \Sigma \models Q$ is not harder than BCQ evaluation over extensional databases (without guarded TGDs).

Guarded TGDs are sufficiently expressive to model the tractable description logic $\mathcal{EL}$ [8, 9] (see Section 9.2), but are still more expressive than actually nec-

essary for modeling *DL-Lite*. Therefore, in Section 4, we consider the further restricted class of *linear TGDs*. These consist of TGDs whose bodies contain only single atoms (and so are trivially guarded, or TGDs whose bodies contain only guards, called *multi-linear TGDs*). Note that this class coincides with the class of inclusion dependencies. A detailed analysis of chase properties of linear TGDs yields the following results.

Theorem: Given a database D, a fixed set of linear TGDs Σ, and a fixed BCQ Q, deciding whether $D \cup \Sigma \models Q$ is in AC_0. In particular, this problem is FO-rewritable, i.e., Q and Σ can be compiled into a first-order formula ϕ such that for each database D, it holds that $D \cup \Sigma \models Q$ iff $D \models \phi$.

In order to capture *DL-Lite*, we further enrich guarded Datalog by two additional other features: negative constraints and keys. A negative constraint is a Horn clause whose body is not necessarily guarded and whose head is the truth constant *false* which we denote by $\bot$. For example, the requirement that a person ID cannot simultaneously appear in the $employee(ID, Name)$ and in the $retired(ID, Name)$ relation can be expressed by:

$$employee(X, Y) \wedge retired(X, Z) \rightarrow \bot\,.$$

While negative constraints do add expressive power to Datalog, they are actually very easy to handle, and we show that the addition of negative constraints does not increase the complexity of query answering. We also allow a limited form of *equality-generating dependencies*, namely, *keys*, to be specified, but we require that these keys be – in a precise sense – not conflicting with the existential rules of the Datalog program. We lift a result from [27] about non-key-conflicting inclusion dependencies to the setting of arbitrary TGDs to prove that the keys that we consider do not increase the complexity. With these additions we have a quite expressive and still extremely efficient version of Datalog$^{\pm}$.

Theorem: Query answering with Datalog$^{\pm}$ based on guarded TGDs (resp., linear TGDs), negative constraints, and keys that do not conflict with the TGDs is possible in polynomial time in the data complexity (resp., FO-rewritable).

Let us refer to the above basic version of Datalog$^{\pm}$ (linear TGDs, negative constraints, and non-conflicting keys) as Datalog$_0^{\pm}$, and to the guarded version with negative constraints and non-conflicting keys as Datalog$_1^{\pm}$. We are finally able to show in Sections 7 to 9 that all description logics of the well-known *DL-Lite* family of description logics [31] smoothly translate into Datalog$_0^{\pm}$. The relationships between Datalog$_0^{\pm}$, Datalog$_1^{\pm}$, *DL-Lite*, and $\mathcal{EL}$ are summarized in Fig. 1.

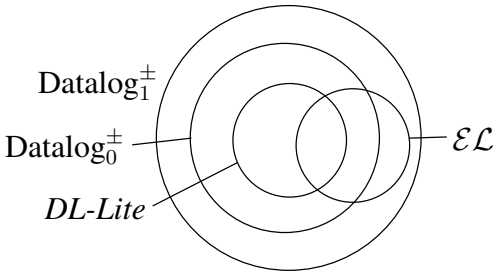

Figure 1: Relationships between Datalog$_0^\pm$, Datalog$_1^\pm$, *DL-Lite*, and $\mathcal{EL}$.

Theorem: The description logics *DL-Lite*$_X$ of the *DL-Lite* family and their extensions with n-ary relations *DLR-Lite*$_X$ can all be reduced to Datalog$_0^\pm$.

Example 2. The axioms of the TBox of Example 1 are translated to the TGDs *ConfPaper*$(X) \rightarrow$ *Article*(X) and *Scientist*$(X) \rightarrow \exists Z$ *isAuthorOf*(X, Z), while the axiom of the ABox is translated to the database atom *Scientist*(*john*).

The translation from the *DL-Lite* family into Datalog$_0^\pm$ is so smooth and natural, that Datalog$_0^\pm$ can rightly be called a DL. Note that Datalog$_0^\pm$ is strictly more expressive than any of the description logics of the *DL-Lite* family. Interestingly, we prove that (at most binary) linear TGDs alone can express useful ontological relationships such as, e.g., $manager(X) \rightarrow manages(X, X)$ that are not expressible in any of the description logics of the *DL-Lite* family.

In the DL community, there is currently a need for enhancing tractable DLs by some nonmonotonic negation (where negative information is derived from the absence of positive one). It was asked whether there is some stratified negation for DLs. Given our translation from *DL-Lite* to Datalog$^\pm$, this amounts to ask whether there is a satisfactory stratified negation for Datalog$^\pm$, and, in particular:

Question 3: Can we extend the concept of safe stratified negation to guarded TGDs?

In classical Datalog with stratified negation [6], each stratum is finite, and the stratum $i + 1$ can be evaluated as soon as all facts in stratum i have been derived. With guarded TGDs, this is not so. Given that usually an infinite number of facts is generated by the chase, each stratum, including the lowest may be infinite, which means that single strata may at no time be fully computed. The difficulty is then, how long to wait before deciding that a negative atom

in a rule body is satisfied. We solve this problem by making use of the above-mentioned constant-depth bounds for atom derivations. We define a new version of the chase that uses a constant-depth bound for establishing whether a negative atom whose arguments all appear in those of a (positive) rule guard is satisfied. We show that this semantics is stratification-independent, corresponds to a perfect model semantics, and that query answering can be done in polynomial time for guarded TGDs and is FO-rewritable for linear TGDs.

The rest of this paper is organized as follows. In Section 2, we give some preliminaries and basic definitions. Sections 3 and 4 deal with guarded Datalog$^{\pm}$ and the special case of linear Datalog$^{\pm}$, respectively. In Section 5, we show how negative constraints can be added. In Section 6, we discuss the addition of keys. Sections 7 to 9 deal with the translation of the *DL-Lite* family to Datalog$^{\pm}$, while Section 10 defines stratified Datalog$^{\pm}$. In Section 11, we discuss related work. Section 12 summarizes the main results and gives an outlook on future research. Note that detailed proofs of all results are given in Appendices A to G.

2. Preliminaries

In this section, we briefly recall some basics on relational databases, conjunctive queries (CQs), Boolean conjunctive queries (BCQs), tuple-generating dependencies (TGDs), and the chase procedure relative to such dependencies.

2.1. Databases and Queries

As for the elementary ingredients, we assume data constants, nulls, and variables as follows; they serve as arguments in atomic formulas in databases, queries, and dependencies. We assume (i) an infinite universe of *data constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{X}$ (used in queries and dependencies). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$.

We next define atomic formulas, which occur in databases, queries, and dependencies, and which are constructed from relation names and terms, as usual. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *relation names* (or *predicate symbols*, or simply *predicates*). A *position* $P[i]$ identifies the i-th argument of a predicate P. A *term* t is a data constant, null, or variable. An *atomic formula*

(or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is n-ary predicate, and $t_1, ..., t_n$ are terms. We denote by $pred(\mathbf{a})$ and $dom(\mathbf{a})$ its predicate and the set of all its arguments, respectively. The latter two notations are naturally extended to sets of atoms and conjunctions of atoms. A conjunction of atoms is often identified with the set of all its atoms.

We are now ready to define the notion of a database relative to a relational schema, as well as the syntax and the semantics of conjunctive and Boolean conjunctive queries to databases. A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. Such D is *ground* iff it contains only atoms with arguments from Δ. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms with the variables $\mathbf{X}$ and $\mathbf{Y}$, and eventually constants, but without nulls. Note that $\Phi(\mathbf{X}, \mathbf{Y})$ may also contain equalities but no inequalities. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$. We often write a BCQ as the set of all its atoms, having constants and variables as arguments, and omitting the quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{X} \to \Delta \cup \Delta_N \cup \mathcal{X}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Example 3. Consider an employee database, which stores information about managers, employees, and departments, where managers may supervise employees and direct departments, and employees may work in a department. The relational schema $\mathcal{R}$ consists of the unary predicates $manager$ and $employee$ as well as the binary predicates $directs$, $supervises$, and $works\_in$ with obvious semantics. A database D for $\mathcal{R}$ is given as follows:

$$D = \{employee(jo), manager(jo), directs(jo, finance), supervises(jo, ada), \\ employee(ada), works\_in(ada, finance)\}\,.$$

It encodes that Jo is an employee and a manager directing the finance department and supervising Ada, who is an employee working in the finance department. A CQ is given by $Q(X) = manager(X) \wedge directs(X, finance)$, which asks for all managers directing the finance department, while a BCQ is given by $Q() = \exists X\, (manager(X) \wedge directs(X, finance))$, often simply abbreviated as the set of

atoms $\{manager(X),\ directs(X,\ finance)\}$, which asks whether there exists a manager directing the finance department. The set of all answers to the former over D is given by $Q(D) = \{jo\}$, while the answer to the latter is *Yes*.

2.2. Tuple-Generating Dependencies (TGDs)

Tuple-generating dependencies (TGDs) describe constraints on databases in the form of generalized Datalog rules with existentially quantified conjunctions of atoms in rule heads; their syntax and semantics are as follows. Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall\mathbf{X}\forall\mathbf{Y}\,\Phi(\mathbf{X},\,\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\Psi(\mathbf{X},\mathbf{Z})$, where $\Phi(\mathbf{X},\,\mathbf{Y})$ and $\Psi(\mathbf{X},\,\mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted $body(\sigma)$ and $head(\sigma)$, respectively. We usually omit the universal quantifiers in TGDs. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X},\mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X},\mathbf{Z})$ to atoms of D. All sets of TGDs are finite here.

Example 4. Consider again the employee database D of Example 3. Some constraints on D along with their encoding as TGDs (where we use "," to denote the Boolean conjunction "$\wedge$") are as follows:

- every manager is an employee:

$$manager(M) \rightarrow employee(M)\,;$$

- every manager directs at least one department:

$$manager(M) \rightarrow \exists P\ directs(M,P)\,;$$

- every employee who directs a department is a manager, and supervises at least another employee who works in the same department:

$$employee(E), directs(E,P) \rightarrow \exists E'\ manager(E), supervises(E,E'),\\ works\_in(E',P)\,;$$

- every employee supervising a manager is a manager:

$$employee(E), supervises(E,E'), manager(E') \rightarrow manager(E)\,.$$

It is not difficult to verify that all the above TGDs are satisfied in D. Consider next the database D' defined as follows:

$$D' = D \cup \{manager(ada)\} .$$

Then, the first and the last TGD listed above are satisfied in D', while the second and the third TGD are not satisfied in D'.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [13], even when the schema and TGDs are fixed [20].

Example 5. Consider again the employee databases D and D' of Examples 3 and 4, respectively, and the set of TGDs Σ of Example 4. Then, D is a model of D and Σ, i.e., $D \in mods(D, \Sigma)$, while D' is not a model of D' and Σ, i.e., $D' \notin mods(D', \Sigma)$, since the second and the third TGD of Example 4 are not satisfied in D'. Trivially, every model of D' and Σ is a superset of D'. In particular, the following databases B_1, B_2, and B_3 are models of D' and Σ:

$$B_1 = D' \cup \{directs(ada, finance), supervises(ada, ada)\} ,$$

$$B_2 = D' \cup \{directs(ada, finance), supervises(ada, bill), works\_in(bill, finance)\} ,$$

$$B_3 = D' \cup \{directs(ada, toy), supervises(ada, bill), works\_in(bill, toy)\} .$$

On the contrary, the following database B_4 is not a model of D' and Σ, since the third TGD of Example 4 is not satisfied in B_4:

$$B_4 = D' \cup \{directs(ada, toy), supervises(ada, tom)\} .$$

Notice that the atom $employee(jo)$ is true in all models of D' and Σ; therefore, the BCQ $\{employee(jo)\}$ evaluates to true over D' and Σ. This also holds for the BCQ $\{directs(ada, X)\}$, while the BCQ $\{directs(ada, finance)\}$ evaluates to false over D' and Σ, since it is false in the database B_3.

We recall that the two problems of CQ and BCQ evaluation under TGDs are LOGSPACE-equivalent [32, 50, 45, 37]. Moreover, it is easy to see that the query output tuple (QOT) problem (as a decision version of CQ evaluation) and BCQ evaluation are AC_0-reducible to each other. Henceforth, we thus focus only on the BCQ evaluation problem. All complexity results carry over to the other problems. We also recall that query answering under TGDs is equivalent to query answering under TGDs with only singleton atoms in their heads. In the sequel, we thus always assume w.l.o.g. that every TGD has a singleton atom in its head.

2.3. The TGD Chase

The *chase* was introduced to enable checking implication of dependencies [65], and later also for checking query containment [50]. It is a procedure for repairing a database relative to a set of dependencies, so that the result of the chase satisfies the dependencies. By "chase", we refer both to the chase procedure and to its output. The TGD chase works on a database through so-called TGD *chase rules* (for an extended chase with also equality-generating dependencies (EGDs), see Section 6). The TGD chase rule comes in two flavors: *restricted* and *oblivious*, where the restricted one applies TGDs only when they are not satisfied (to repair them), while the oblivious one always applies TGDs (if they produce a new result). We focus on the oblivious one, since it makes proofs technically simpler. The *(oblivious) TGD chase rule* defined below is the building block of the chase.

TGD CHASE RULE. Consider a database D for a relational schema $\mathcal{R}$, and a TGD σ on $\mathcal{R}$ of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D. Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the atom $h_1(\Psi(\mathbf{X}, \mathbf{Z}))$ if not already in D (which is possible when $\mathbf{Z}$ is empty). ■

The chase algorithm for a database D and a set of TGDs Σ consists of an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which leads as result to a (possibly infinite) chase for D and Σ. Formally, the *chase of level up to* 0 of D relative to Σ, denoted $chase^0(D, \Sigma)$, is defined as D, assigning to every atom in D the *(derivation) level* 0. For every $k \geqslant 1$, the *chase of level up to* k of D relative to Σ, denoted $chase^k(D, \Sigma)$, is constructed as follows: let $I_1, \ldots, I_n$ be all possible images of bodies of TGDs in Σ relative to some homomorphism such that (i) $I_1, \ldots, I_n \subseteq chase^{k-1}(D, \Sigma)$ and

(ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding TGD application on $chase^{k-1}(D,\Sigma)$, choosing the applied TGDs and homomorphisms in a linear and lexicographic order, respectively, and assigning to every new atom the *(derivation) level* k. The *chase* of D relative to Σ, denoted $chase(D,\Sigma)$, is then defined as the limit of $chase^k(D,\Sigma)$ for $k \to \infty$.

The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $chase(D,\Sigma)$ onto every $B \in mods(D,\Sigma)$ [37, 20]. This result implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $chase(D,\Sigma) \models Q$.

Example 6. Consider again the employee database D' of Example 3 and the set of TGDs Σ of Example 4. Then, in the construction of $chase(D',\Sigma)$, we apply first the second TGD of Example 4 to $manager(jo)$ (resp., $manager(ada)$), adding $directs(jo,z_1)$ (resp., $directs(ada,z_2)$), where z_1 and z_2 are "fresh" nulls, and then the third TGD to $employee(jo)$ and $directs(jo,z_1)$ (resp., $employee(ada)$ and $directs(ada,z_2)$), adding $supervises(jo,z_3)$ and $works\_in(z_3,z_1)$ (resp., $supervises(ada,z_4)$ and $works\_in(z_4,z_2)$), where z_3 and z_4 are "fresh" nulls. Hence, the construction yields a finite chase $chase(D',\Sigma)$, given as follows:

$$\begin{aligned} chase(D',\Sigma) = D' \cup \{ & directs(jo,z_1), directs(ada,z_2), \\ & supervises(jo,z_3), works\_in(z_3,z_1), \\ & supervises(ada,z_4), works\_in(z_4,z_2) \}. \end{aligned}$$

Here, every atom in D' is of level 0, the two atoms $directs(jo,z_1)$ and $directs(ada,z_2)$ are of level 1, and the other four atoms are of level 2.

3. Guarded Datalog$^{\pm}$

We now introduce guarded Datalog$^{\pm}$ as a class of special TGDs that exhibit computational tractability in the data, while being at the same time expressive enough to model ontologies. BCQs relative to such TGDs can be evaluated on a finite part of the chase, which is of constant size when the query and the TGDs are fixed. Based on this result, the data complexity of evaluating BCQs relative to guarded TGDs turns out to be polynomial in general and linear for atomic queries.

A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ. The non-guard atoms in the body of σ are the *side atoms* of σ.

Example 7. The TGD $r(X,Y), s(Y,X,Z) \rightarrow \exists W\, s(Z,X,W)$ is guarded (where $s(Y,X,Z)$ is the guard, and $r(X,Y)$ is a side atom), while the TGD $r(X,Y)$, $r(Y,Z) \rightarrow r(X,Z)$ is not guarded, since it has no guard, i.e., no body atom contains all the (universally quantified) variables in the body. Furthermore, it is easy to verify that every TGD in Example 4 is guarded.

Note that sets of guarded TGDs (with single-atom heads) are theories in the guarded fragment of first-order logic [4]. Note also that guardedness is a truly fundamental class ensuring decidability. As shown in [20], adding a single unguarded Datalog rule to a guarded Datalog$^{\pm}$ program may destroy decidability.

In the sequel, let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. We first give some preliminary definitions as follows. The *chase graph* for D and Σ is the directed graph consisting of $chase(D,\Sigma)$ as the set of nodes and having an arrow from $\mathbf{a}$ to $\mathbf{b}$ iff $\mathbf{b}$ is obtained from $\mathbf{a}$ and possibly other atoms by a one-step application of a TGD $\sigma \in \Sigma$. Here, we mark $\mathbf{a}$ as *guard* iff $\mathbf{a}$ is the guard of σ. As a subgraph of the chase graph for D and Σ, the *guarded chase forest* for D and Σ contains (i) all atoms $\mathbf{a} \in D$ as nodes and (ii) any two atoms $\mathbf{a}, \mathbf{b} \in chase(D,\Sigma)$ as nodes along with an arrow from $\mathbf{a}$ to $\mathbf{b}$ iff $\mathbf{b}$ is obtained from $\mathbf{a}$ and possibly other atoms by a one-step application of a TGD $\sigma \in \Sigma$ with $\mathbf{a}$ as guard. The *subtree* of an atom $\mathbf{a}$ in this forest, denoted $subtree(\mathbf{a})$, is the restriction of the forest to all descendants of $\mathbf{a}$. The *type* of an atom $\mathbf{a}$, denoted $type(\mathbf{a})$, is the set of all atoms $\mathbf{b}$ in $chase(D,\Sigma)$ that have only constants and nulls from $\mathbf{a}$ as arguments. Note that the subtree of $\mathbf{a}$ in the guarded chase forest depends only on the set of all atoms in the type of $\mathbf{a}$ (and no others).

Example 8. Consider the database $D = \{r(a,b), s(b)\}$ and the set of TGDs Σ consisting of the following two TGDs σ_1 and σ_2:

$$\begin{array}{lrcl} \sigma_1\colon & r(X,Y), s(Y) & \rightarrow & \exists Z\, r(Z,X)\,, \\ \sigma_2\colon & r(X,Y) & \rightarrow & s(X)\,. \end{array}$$

The first part of the (infinite) chase graph (resp., guarded chase forest) for D and Σ is shown in Fig. 2, left (resp., right) side, where the arrows have the applied TGDs as labels (formally not a part of the graph (resp., forest)). The number on the upper right side of every atom indicates the derivation level of the atom. The subtree of $r(z_1,a)$ in the guarded chase forest is also shown in Fig. 2, right side. Note also that the type of $r(z_1,a)$ consists of the atoms $r(z_1,a)$, $s(a)$, and $s(z_1)$.

Given a finite set $S \subseteq \Delta \cup \Delta_N$, two sets of atoms A_1 and A_2 are *S-isomorphic* (or *isomorphic* if $S = \emptyset$) iff a bijection $\beta\colon A_1 \cup dom(A_1) \rightarrow A_2 \cup dom(A_2)$ exists

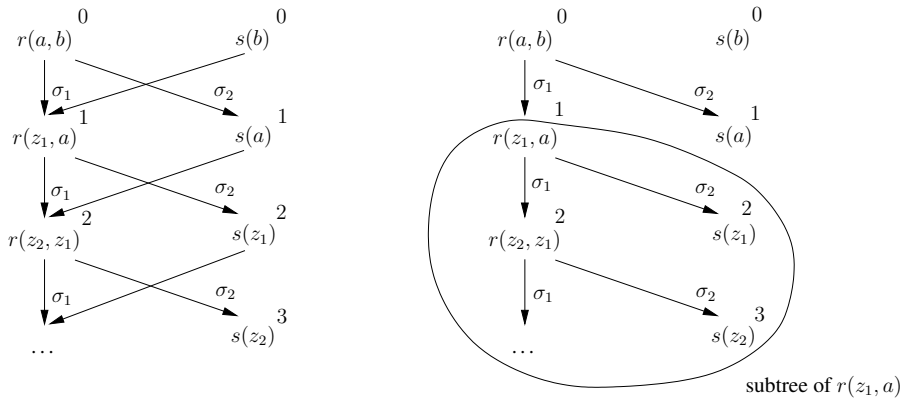

Figure 2: Chase graph (left side) and guarded chase forest (right side) for Example 8.

such that (i) β and β^{-1} are homomorphisms, and (ii) $\beta(c) = c = \beta^{-1}(c)$ for all $c \in S$. Two atoms a_1 and a_2 are *S-isomorphic* (or *isomorphic* if $S = \emptyset$) iff $\{a_1\}$ and $\{a_2\}$ are S-isomorphic. The notion of S-isomorphism (or isomorphism if $S = \emptyset$) is naturally extended to more complex structures, such as pairs of two subtrees (V_1, E_1) and (V_2, E_2) of the guarded chase forest, and two pairs $(\mathbf{b}_1, S_1)$ and $(\mathbf{b}_2, S_2)$, where $\mathbf{b}_1$ and $\mathbf{b}_2$ are atoms, and S_1 and S_2 are sets of atoms.

Example 9. The two sets of atoms $\{\mathbf{a}_1 \colon r(a, z_1, z_2), \mathbf{a}_2 \colon s(b, z_2), \mathbf{a}_3 \colon t(z_3, z_4\}$ and $\{\mathbf{b}_1 \colon r(a, z_1, z_5), \mathbf{b}_2 \colon s(b, z_5), \mathbf{b}_3 \colon t(z_6, z_7)\}$, where $a, b \in \Delta$ and $z_1, \ldots, z_7 \in \Delta_N$, are $\{a, z_1\}$-isomorphic via the bijection β defined by $\beta(\mathbf{a}_i) = \mathbf{b}_i,\ i \in \{1, 2, 3\}$, $\beta(z_2) = z_5$, $\beta(z_3) = z_6$, $\beta(z_4) = z_7$, and $\beta(c) = c$ for all other $c \in \Delta \cup \Delta_N$. Furthermore, let $\mathbf{a} = r(a, b, z_1)$, where $a, b \in \Delta$ and $z_1 \in \Delta_N$. Then, $s(b, z_3)$ and $s(b, z_4)$ are $dom(\mathbf{a})$-isomorphic, while $s(b, z_3)$ and $s(b, z_1)$ are not (with $z_3, z_4 \in \Delta_N$).

The following lemma shows that if two atoms in the guarded chase forest for D and Σ have S-isomorphic types, then their subtrees are also S-isomorphic, which can be proved by induction on the number of applications of the TGD chase rule to generate the subtrees of the two atoms.

Lemma 1. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. Let S be a finite set of constants and nulls from $\Delta \cup \Delta_N$, and let $\mathbf{a}_1$ and $\mathbf{a}_2$ be atoms from $chase(D, \Sigma)$ with S-isomorphic types. Then, the subtree of $\mathbf{a}_1$ is S-isomorphic to the subtree of $\mathbf{a}_2$.*

The next lemma provides, given an atom $\mathbf{a} \in chase(D, \Sigma)$, an upper bound for the number of all non-$dom(\mathbf{a})$-isomorphic pairs consisting of an atom and

a type with arguments from $\mathbf{a}$ and new nulls. The result follows from a simple combinatorial analysis of the number of all possible such pairs.

Lemma 2. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. Let w be the maximal arity of a predicate in $\mathcal{R}$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and $\mathbf{a} \in chase(D, \Sigma)$. Let P be a set of pairs $(\mathbf{b}, S)$, each consisting of an atom $\mathbf{b}$ and a type S of atoms $\mathbf{c}$ with arguments from $\mathbf{a}$ and new nulls. If $|P| > \delta$, then P contains at least two $dom(\mathbf{a})$-isomorphic pairs.*

We next define the guarded depth of atoms in the guarded chase forest as follows. The *guarded depth* of an atom $\mathbf{a}$ in the guarded chase forest for D and Σ, denoted $depth(\mathbf{a})$, is the length of the path from D to $\mathbf{a}$ in the forest. Note that this is in general different from the derivation level in the chase (see Example 10). The *guarded chase of level up to $k \geqslant 0$* for D and Σ, denoted $g\text{-}chase^k(D, \Sigma)$, is the set of all atoms in the guarded chase forest of depth at most k.

Example 10. Consider the database $D = \{r_1(a, b)\}$ and the set of TGDs Σ consisting of the following three TGDs σ_1, σ_2, and σ_3:

$$\begin{array}{lrcl} \sigma_1\colon & r_3(X,Y) & \rightarrow & r_2(X)\,, \\ \sigma_2\colon & r_1(X,Y) & \rightarrow & \exists Z\, r_3(Y,Z)\,, \\ \sigma_3\colon & r_1(X,Y), r_2(Y) & \rightarrow & r_1(Y,X)\,. \end{array}$$

The chase graph for D and Σ is shown in Fig. 3. It nearly coincides with the guarded chase forest for D and Σ, where only the dashed arrow is removed. Every atom is also labeled with its guarded depth and its derivation level.

The following key lemma shows that for each set of guarded TGDs Σ, there exists a constant k such that for every database D and every atom $\mathbf{a}$ generated at some depth level d while chasing D with Σ, such that whenever the same chase generates an atom $\mathbf{b}$ whose arguments are among those of $\mathbf{a}$, then $\mathbf{b}$ must be generated at depth at most $d + k$. The main idea behind the proof by contradiction, using the above Lemmas 1 and 2, is that the subtree of an atom $\mathbf{a}$ depends only on $\mathbf{a}$ and the type of $\mathbf{a}$, and the number of non-$dom(\mathbf{a})$-isomorphic pairs consisting of an atom and its type is bounded by a constant, depending only on $\mathcal{R}$.

Lemma 3. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. Let $\mathbf{a}$ be a guard in the chase graph for D and Σ, and let $\mathbf{b} \in type(\mathbf{a})$. Then, $depth(\mathbf{b}) \leqslant depth(\mathbf{a}) + k$, where k depends only on $\mathcal{R}$.*

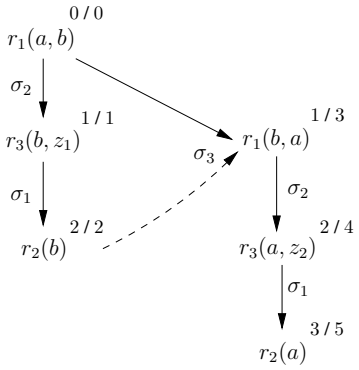

Figure 3: Guarded depth / derivation level of atoms in the chase graph.

The following lemma shows that BCQs Q can be evaluated using only a finite, initial portion of the guarded chase forest, whose size depends only on the query Q and the relational schema $\mathcal{R}$. The result is proved similarly to Lemma 3, showing that every path from D to (the image of) a query atom in the guarded chase forest, whose length exceeds a certain value (depending on Q and $\mathcal{R}$), has two atoms with $dom(\mathbf{a})$-isomorphic subtrees (since two atoms and their types are $dom(\mathbf{a})$-isomorphic), and thus Q can also be evaluated "closer" to D.

Lemma 4. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a set of guarded TGDs on $\mathcal{R}$, and Q be a BCQ over $\mathcal{R}$. If there exists a homomorphism μ that maps Q into $chase(D, \Sigma)$, then there exists a homomorphism λ that maps Q into $g\text{-}chase^k(D, \Sigma)$, where k depends only on Q and $\mathcal{R}$.*

Intuitively, the chase of a database relative to a set of guarded TGDs has a "periodicity" of atoms and their types, as illustrated by the following example.

Example 11. Every derivation level $k \geqslant 2$ of the chase graph for Example 8 in Fig. 8 is given by two atoms $r(z_k, z_{k-1})$ and $s(z_{k-1})$, where the type of the former is given by the three atoms $r(z_k, z_{k-1})$, $s(z_{k-1})$, and $s(z_k)$. This "pattern" repeats indefinitely in the chase, as easily seen. For example, a BCQ $Q = \{r(X, Y), r(Z, X), r(W, Z)\}$ will necessarily map onto three atoms that form a path in the guarded chase forest: however deep these atoms are in the chase, Q can anyway also be mapped onto the first levels, e.g., onto $\{r(z_2, z_1), r(z_3, z_2), r(z_4, z_3)\}$.

The above lemma informally says that whenever (homomorphic images of) the query atoms are contained in the chase, then they are also contained in a finite,

initial portion of the guarded chase forest, whose size is determined only by the query and the schema. However, it does not yet ensure that also the whole derivation of the query atoms are contained in such a portion of the forest. This slightly stronger property is captured by the following definition.

Definition 1. Let $\mathcal{R}$ be a relational schema, and Σ be a set of TGDs on $\mathcal{R}$. Then, we say that Σ has the *bounded guard-depth property (BGDP)* iff, for every database D for $\mathcal{R}$ and for every BCQ Q, whenever there exists a homomorphism μ that maps Q into $chase(D, \Sigma)$, then there exists a homomorphism λ of this kind such that all ancestors of $\lambda(Q)$ in the chase graph for D and Σ are contained in $g\text{-}chase^{\gamma_g}(D, \Sigma)$, where γ_g depends only on Q and $\mathcal{R}$.

The next theorem shows that, in fact, guarded TGDs have also this stronger bounded guard-depth property. The proof of this result is based on the above Lemmas 3 and 4, where the former now also assures that all side atoms that are necessary in the derivation of the query atoms are contained in a finite, initial portion of the guarded chase forest, whose size is determined only by Q and $\mathcal{R}$ (which is slightly larger than the one for the query atoms only).

Theorem 5. *Guarded TGDs enjoy the BGDP.*

By this theorem, deciding BCQs in the guarded case is in P in the data complexity (where all but the database is fixed) [20]. It is also hard for P, as can be proved by reduction from propositional logic programming.

Theorem 6. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a set of guarded TGDs on $\mathcal{R}$, and Q be a BCQ over $\mathcal{R}$. Then, deciding $D \cup \Sigma \models Q$ is P-complete in the data complexity.*

Deciding Boolean atomic queries in the guarded case can even be done in linear time in the data complexity, as the following theorem shows, which holds by a reduction to propositional logic programming. Note that since general BCQs are not necessarily guarded, they are in general not reducible to atomic queries.

Theorem 7. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a set of guarded TGDs on $\mathcal{R}$, and Q be a Boolean atomic query over $\mathcal{R}$. Then, deciding $D \cup \Sigma \models Q$ can be done in linear time in the data complexity.*

4. Linear Datalog$^{\pm}$

We now introduce linear Datalog$^{\pm}$ as a variant of guarded Datalog$^{\pm}$, where query answering is even FO-rewritable in the data complexity. Nonetheless, (an extension of) linear Datalog$^{\pm}$ is still expressive enough for representing ontologies, as we will show in Sections 7 and 8 for ontologies encoded in the description logics of the *DL-Lite* family (*DL-Lite*$_{\mathcal{F}}$, *DL-Lite*$_{\mathcal{R}}$, and *DL-Lite*$_{\mathcal{A}}$ [31, 72]).

A TGD is *linear* iff it contains only a singleton body atom (i.e., the TGD is of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is an atom).

Example 12. Consider again the TGDs of Example 4. As easily verified, the first two are linear, while the last two are not. Another linear TGD is $directs(E, P) \rightarrow employee(E)$, restricting the first argument of the directs relation to employees.

Observe that linear Datalog$^{\pm}$ generalizes the well-known class of *inclusion dependencies*, and that this generalization is strict, for example, the linear TGD *supervises*$(X, X) \rightarrow$ *manager*(X), which asserts that all people supervising themselves are managers, is not expressible with inclusion dependencies.

We next define the bounded derivation-depth property for sets of TGDs, which is strictly stronger than the bounded guard-depth property (see Definition 1), since the former implies the latter, but not vice versa. Informally, the bounded derivation-depth property says that whenever (homomorphic images of) the query atoms are contained in the chase, then they (along with their derivations) are also contained in a finite, initial portion of the chase graph (rather than the guarded chase forest), whose size depends only on the query and the schema.

Definition 2. Let $\mathcal{R}$ be a relational schema, and Σ be a set of TGDs on $\mathcal{R}$. Then, we say that Σ has the *bounded derivation-depth property (BDDP)* iff, for every database D for $\mathcal{R}$ and for every BCQ Q over $\mathcal{R}$, whenever $D \cup \Sigma \models Q$, then $chase^{\gamma_d}(D, \Sigma) \models Q$, where γ_d depends only on Q and $\mathcal{R}$.

Clearly, in the case of linear TGDs, for every $\mathbf{a} \in chase(D, \Sigma)$, the subtree of $\mathbf{a}$ is now determined only by $\mathbf{a}$ itself, while in the case of guarded TGDs it depends on $type(\mathbf{a})$. Therefore, for a single atom, its depth coincides with the number of applications of the TGD chase rule that are necessary to generate it. That is, the guarded chase forest coincides with the chase graph. Thus, by Theorem 5, we immediately obtain that linear TGDs have the bounded derivation-depth property.

Corollary 8. *Linear TGDs enjoy the BDDP.*

We next recall the notion of first-order rewritability for classes of TGDs. A class of TGDs $\mathcal{C}$ is *first-order rewritable* (or *FO-rewritable*) iff for every set of TGDs Σ in $\mathcal{C}$ and for every BCQ Q, there exists a first-order query Q_Σ such that, for every database D, it holds that $D \cup \Sigma \models Q$ iff $D \models Q_\Sigma$. Since answering first-order queries is in AC_0 in the data complexity [81], also BCQ answering under FO-rewritable TGDs is in AC_0 in the data complexity.

The following result shows that BCQs Q relative to TGDs Σ with the bounded derivation-depth property are FO-rewritable. The main ideas behind its proof are informally summarized as follows. Since the derivation depth and the number of body atoms in TGDs in Σ are bounded, the number of all database ancestors of query atoms is also bounded. So, the number of all non-isomorphic sets of potential database ancestors with variables as arguments is also bounded. Take the existentially quantified conjunction of every such ancestor set where Q is answered positively. Then, the FO-rewriting of Q is the disjunction of all these formulas.

Theorem 9. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a set of TGDs on $\mathcal{R}$, and Q be a BCQ over $\mathcal{R}$. If Σ enjoys the BDDP, then Q is FO-rewritable.*

As an immediate consequence of Corollary 8 and Theorem 9, we obtain that BCQs are FO-rewritable in the linear case.

Corollary 10. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a set of linear TGDs on $\mathcal{R}$, and Q be a BCQ over $\mathcal{R}$. Then, Q is FO-rewritable.*

Observe that all the above results also apply to *multi-linear* TGDs, which are TGDs with only guards in their bodies, since here the guarded chase forest can be chosen in such a way that the depth of all its atoms coincides with their derivation depth. Formally, a TGD σ is *multi-linear* iff all its body atoms have the same variables (i.e., σ has the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms $p_i(\mathbf{X}, \mathbf{Y})$, each containing each variable of $\mathbf{X}$ and $\mathbf{Y}$).

5. Adding (Negative) Constraints

In this section, we extend Datalog$^\pm$ by (negative) constraints, which are an important ingredient, in particular, for representing ontologies.

A *negative constraint* (or simply *constraint*) is a first-order formula of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ is a (not necessarily guarded) conjunction of atoms. It is often also written as $\forall \mathbf{X}\, \Phi'(\mathbf{X}) \rightarrow \neg p(\mathbf{X})$, where $\Phi'(\mathbf{X})$ is obtained

from $\Phi(\mathbf{X})$ by removing the atom $p(\mathbf{X})$. We usually omit the universal quantifiers, and we implicitly assume that all sets of constraints are finite here.

Example 13. If the two unary predicates c and c' represent two classes (also called concepts in DLs), we may use the constraint $c(X), c'(X) \rightarrow \bot$ to assert that the two classes have no common instances. Similarly, if additionally the binary predicate r represents a relationship (also called a role in DLs), we may use $c(X), r(X, Y) \rightarrow \bot$ to enforce that no member of the class c participates to the relationship r. Furthermore, if the two binary predicates r and r' represent two relationships, we may use the constraint $r(X, Y), r'(X, Y) \rightarrow \bot$ to express that the two relationships are disjoint.

Query answering on a database D under a set of TGDs Σ_T (as well as a set of EGDs Σ_E as introduced in the next section) and a set of constraints Σ_C can be done effortless by additionally checking that every constraint $\sigma = \Phi(\mathbf{X}) \rightarrow \bot \in \Sigma_C$ is satisfied in D and Σ_T, each of which can be done by checking that the BCQ $Q_\sigma = \Phi(\mathbf{X})$ evaluates to false on D and Σ_T. We write $D \cup \Sigma_T \models \Sigma_C$ iff every $\sigma \in \Sigma_C$ is false in D and Σ_T. We thus obtain immediately the following result. Here, a BCQ Q is *true* in D and Σ_T and Σ_C, denoted $D \cup \Sigma_T \cup \Sigma_C \models Q$, iff (i) $D \cup \Sigma_T \models Q$ or (ii) $D \cup \Sigma_T \not\models \Sigma_C$ (as usual in DLs).

Theorem 11. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ_T and Σ_C be sets of TGDs and constraints on $\mathcal{R}$, respectively, and Q be a BCQ on $\mathcal{R}$. Then, $D \cup \Sigma_T \cup \Sigma_C \models Q$ iff (i) $D \cup \Sigma_T \models Q$ or (ii) $D \cup \Sigma_T \models Q_\sigma$ for some $\sigma \in \Sigma_C$.*

As an immediate consequence, we obtain that constraints do not increase the data complexity of answering BCQs in the guarded (resp., linear) case.

Corollary 12. *Answering BCQs on databases under guarded (resp., linear) TGDs and constraints has the same data complexity as answering BCQs on databases under guarded (resp., linear) TGDs alone.*

6. Adding Equality-Generating Dependencies (EGDs) and Keys

In this section, we add equality-generating dependencies (EGDs) to guarded (and linear) Datalog$^{\pm}$, which are also important when representing ontologies. Note that EGDs generalize *functional dependencies* (*FDs*) and, in particular, *key dependencies* (or *keys*) [1]. In *DL-Lite* (see Sections 7 and 8), general EGDs cannot be formulated, but only keys. Therefore, we mainly focus on keys here.

We transfer a result by [27] about *non-key-conflicting (NKC)* inclusion dependencies to the more general setting of guarded Datalog$^{\pm}$.

However, while adding negative constraints is effortless from a computational perspective, adding EGDs is more problematic: The interaction of TGDs and EGDs leads to undecidability of query answering even in simple cases, such that of functional and inclusion dependencies [33], or keys and inclusion dependencies (see, e.g., [27], where the proof of undecidability is done in the style of Vardi as in [50]). It can even be seen that a *fixed* set of EGDs and guarded TGDs can simulate a universal Turing machine, and thus query answering and even propositional ground atom inference is undecidable for such dependencies. For this reason, we consider a restricted class of EGDs, namely, *non-conflicting key dependencies* (or *NC keys*), which show a controlled interaction with TGDs (and negative constraints), such that they do not increase the complexity of answering BCQs. Nonetheless, this class is sufficient for modeling ontologies.

An *equality-generating dependency* (or *EGD*) σ is a first-order formula of the form $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $body(\sigma)$, is a (not necessarily guarded) conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. We call $X_i = X_j$ the *head* of σ, denoted $head(\sigma)$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. We usually omit the universal quantifiers in EGDs, and all sets of EGDs are finite here.

Example 14. The following formula σ is an equality-generating dependency:

$$r_1(X, Y), r_2(Y, Z) \rightarrow Y = Z.$$

The database $D = \{r_1(a, b), r_2(b, b)\}$ satisfies σ, because every homomorphism h mapping the body of σ to D is such that $h(Y) = h(Z)$. On the contrary, the database $D = \{r_1(a, b), r_2(b, c)\}$ does not satisfy σ.

An EGD σ on $\mathcal{R}$ of the form $\Phi(\mathbf{X}) \rightarrow X_i = X_j$ is *applicable* to a database D for $\mathcal{R}$ iff there exists a homomorphism $\eta\colon \Phi(\mathbf{X}) \rightarrow D$ such that $\eta(X_i)$ and $\eta(X_j)$ are different and not both constants. If $\eta(X_i)$ and $\eta(X_j)$ are different constants in Δ, then there is a *hard violation* of σ, and the chase *fails*. Otherwise, the result of the application of σ to D is the database $h(D)$ obtained from D by replacing every occurrence of a non-constant element $e \in \{\eta(X_i), \eta(X_j)\}$ in D by the other element e' (if e and e' are both nulls, then e precedes e' in the lexicographic order). Note that h is a homomorphism, but not necessarily an endomorphism of D,

since $h(D)$ is not necessarily a subset of D. But for the special class of TGDs and EGDs that we define in this section, h is actually an endomorphism of D.

The *chase* of a database D, in the presence of two sets Σ_T and Σ_E of TGDs and EGDs, respectively, denoted $chase(D, \Sigma_T \cup \Sigma_E)$, is computed by iteratively applying (1) a single TGD once, according to the standard order and (2) the EGDs, as long as they are applicable (i.e., until a fixpoint is reached).

Example 15. Consider the following set of TGDs and EGDs $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$:

$$\begin{array}{lrcl} \sigma_1: & r(X,Y) & \rightarrow & \exists Z\, s(X,Y,Z), \\ \sigma_2: & s(X,Y,Z) & \rightarrow & Y = Z, \\ \sigma_3: & r(X,Y), s(Z,Y,Y) & \rightarrow & X = Y. \end{array}$$

Let D be the database $\{r(a,b)\}$. In the computation of $chase(D,\Sigma)$, we first apply σ_1 and add the fact $s(a,b,z_1)$, where z_1 is a null. Then, the application of σ_2 on $s(a,b,z_1)$ yields $z_1 = b$, thus turning $s(a,b,z_1)$ into $s(a,b,b)$. Now, we apply σ_3 on $r(a,b)$ and $s(a,b,b)$, and by equating $a = b$, the chase fails; this is a hard violation, since both a and b are constants in Δ.

6.1. Separability

We now first focus on the semantic notion of separability for EGDs, which formulates a controlled interaction of EGDs and TGDs / (negative) constraints, so that the EGDs do not increase the complexity of answering BCQs.

Definition 3. Let $\mathcal{R}$ be a relational schema, and Σ_T and Σ_E be sets of TGDs and EGDs on $\mathcal{R}$, respectively. Then, Σ_E is *separable* from Σ_T iff for every database D for $\mathcal{R}$, the following conditions (i) and (ii) are both satisfied:

(i) If there is a hard violation of an EGD of Σ_E in $chase(D, \Sigma_T \cup \Sigma_E)$, then there is also a hard violation of some EGD of Σ_E in D.
(ii) If there is no chase failure, then for every BCQ Q, it holds that $chase(D, \Sigma_T \cup \Sigma_E) \models Q$ iff $chase(D, \Sigma_T) \models Q$.

The following result shows that adding separable EGDs to TGDs and constraints does not increase the data complexity of answering BCQs in the guarded and linear case. It follows immediately from the fact that the separability of EGDs implies that chase failure can be directly evaluated on D.

Theorem 13. *Let $\mathcal{R}$ be a relational schema, Σ_T and Σ_E be fixed sets of TGDs and EGDs on $\mathcal{R}$, respectively, where Σ_E is separable from Σ_T, and Σ_C be a fixed set of constraints on $\mathcal{R}$. Let $\mathcal{Q}_C$ be the disjunction of all Q_σ with $\sigma \in \Sigma_C$. Then:*

(a) If deciding $D \cup \Sigma_T \models Q \vee \mathcal{Q}_C$ is feasible in polynomial time for each fixed query Q, then so is deciding $D \cup \Sigma_T \cup \Sigma_E \models Q \vee \mathcal{Q}_C$.
(b) If deciding $D \cup \Sigma_T \models Q \vee \mathcal{Q}_C$ is FO-rewritable for each fixed query Q, then so is deciding $D \cup \Sigma_T \cup \Sigma_E \models Q \vee \mathcal{Q}_C$.

6.2. Non-Conflicting Keys

We next provide a sufficient syntactic condition for the separability of EGDs. We assume that the reader is familiar with the notions of a functional dependency (FD) (which informally encodes that certain attributes of a relation functionally depend on others) and a key (dependency) (which is informally a tuple-identifying set of attributes of a relation) [1]. Clearly, FDs are special types of EGDs. A key κ of a relation r can be written as a set of FDs that specify that κ determines each other attribute of r. Thus, keys can be identified with sets of EGDs. It will be clear from the context when we regard a key as a set of attribute positions, and when we regard it as a set of EGDs. The following definition generalizes the notion of "non-key-conflicting" dependency relative to a set of keys, introduced in [27], to the context of arbitrary TGDs.

Definition 4. Let κ be a key, and σ be a TGD of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, r(\mathbf{X}, \mathbf{Z})$. Then, κ is *non-conflicting (NC)* with σ iff either (i) the relational predicate on which κ is defined is different from r, or (ii) the positions of κ in r are not a proper subset of the $\mathbf{X}$-positions in r in the head of σ, and every variable in $\mathbf{Z}$ appears only once in the head of σ. We say κ is *non-conflicting (NC)* with a set of TGDs Σ_T iff κ is NC with every $\sigma \in \Sigma_T$. A set of keys Σ_K is *non-conflicting (NC)* with Σ_T iff every $\kappa \in \Sigma_K$ is NC with Σ_T.

Example 16. Consider the four keys κ_1, κ_2, κ_3, and κ_4 defined by the key attribute sets $\mathbf{K}_1 = \{r[1], r[2]\}$, $\mathbf{K}_2 = \{r[1], r[3]\}$, $\mathbf{K}_3 = \{r[3]\}$, and $\mathbf{K}_4 = \{r[1]\}$, respectively, and the TGD $\sigma = p(X, Y) \rightarrow \exists Z\, r(X, Y, Z)$. Then, the head predicate of σ is r, and the set of positions in r with universally quantified variables is $\mathbf{H} = \{r[1], r[2]\}$. Observe that all keys but κ_4 are NC with σ, since only $\mathbf{K}_4 \subset \mathbf{H}$. Roughly, every atom added in a chase by applying σ would have a fresh null in some position in $\mathbf{K}_1$, $\mathbf{K}_2$, and $\mathbf{K}_3$, thus never firing κ_1, κ_2, and κ_3, respectively.

The following theorem shows that the property of being NC between keys and TGDs implies their separability. This generalizes a useful result of [27] on inclusion dependencies to the much larger class of all TGDs. The main idea behind the proof can be roughly described as follows. The NC condition between a key κ and

a TGD σ assures that either (a) the application of σ in the chase generates an atom with a fresh null in a position of κ, and so the fact does not violate κ (see also Example 16), or (b) the $\mathbf{X}$-positions in the predicate r in the head of σ coincide with the key positions of κ in r, and thus any newly generated atom must have fresh distinct nulls in all but the key position, and may eventually be eliminated without violation. It then follows that the full chase does not fail. Since the new nulls are all distinct, it also contains a homomorphic image of the TGD chase. Therefore, the full chase is in fact homomorphically equivalent to the TGD chase.

Theorem 14. *Let $\mathcal{R}$ be a relational schema, Σ_T and Σ_K be sets of TGDs and keys on $\mathcal{R}$, respectively, such that Σ_K is NC with Σ_T. Then, Σ_K is separable from Σ_T.*

We conclude this section by stating that in the NC case, keys do not increase the data complexity of answering BCQs under guarded (resp., linear) TGDs and constraints. This result follows immediately from Theorems 14 and 13.

Corollary 15. *Let $\mathcal{R}$ be a relational schema, Σ_T and Σ_K be fixed sets of TGDs and keys on $\mathcal{R}$, respectively, where Σ_K is NC with Σ_T, and Σ_C be a fixed set of constraints on $\mathcal{R}$. Let $\mathcal{Q}_C$ be the disjunction of all Q_σ such that $\sigma \in \Sigma_C$. Then:*

(a) If Σ_T are guarded TGDs, then deciding $D \cup \Sigma_T \cup \Sigma_K \models Q \vee \mathcal{Q}_C$ is feasible in polynomial time.

(b) If Σ_T are linear TGDs, then deciding $D \cup \Sigma_T \cup \Sigma_K \models Q \vee \mathcal{Q}_C$ is FO-rewritable.

7. Ontology Querying in *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$

In this section, we show that the description logics *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$ of the *DL-Lite* family [31] can both be reduced to linear (or multi-linear) Datalog$^\pm$ with (negative) constraints and NC keys, called Datalog$_0^\pm$, and that the former are strictly less expressive than the latter. More specifically, we show how Datalog$_0^\pm$ can be used for answering BCQs in *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$ ontologies. We first recall the syntax and the semantics of *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$. We then define the translation and provide the representation and expressivity results.

Note that *DL-Lite*$_\mathcal{R}$ is able to fully capture the (DL fragment of) RDF Schema [16], the vocabulary description language for RDF; see [36] for a translation. Hence, Datalog$_0^\pm$ is also able to fully capture (the DL fragment of) RDF Schema.

The other description logics of the *DL-Lite* family [31] can be similarly translated into Datalog$_0^\pm$: the translation of *DL-Lite*$_\mathcal{A}$ into Datalog$_0^\pm$ is given in Section 8, and the translations for the other DLs are sketched in Section 9. Note

that it is mainly for didactic reasons that we start with the simpler *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$, and we continue with the slightly more complex *DL-Lite*$_{\mathcal{A}}$.

Intuitively, DLs model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations on classes of individuals, respectively. A DL knowledge base (or ontology) encodes in particular subset relationships between concepts, subset relationships between roles, the membership of individuals to concepts, the membership of pairs of individuals to roles, and functional dependencies on roles.

7.1. Syntax of DL-Lite$_{\mathcal{F}}$ *and DL-Lite*$_{\mathcal{R}}$

We now recall the syntax of *DL-Lite*$_{\mathcal{F}}$ (also simply called *DL-Lite*). As for the elementary ingredients, we assume pairwise disjoint sets of *atomic concepts*, *abstract roles*, and *individuals* $\mathbf{A}$, $\mathbf{R}_A$, and $\mathbf{I}$, respectively.

These elementary ingredients are used to construct roles and concepts, which are defined as follows: A *basic role* Q is either an atomic role $P \in \mathbf{R}_A$ or its inverse P^-. A *(general) role* R is either a basic role Q or the negation of a basic role $\neg Q$. A *basic concept* B is either an atomic concept $A \in \mathbf{A}$ or an existential restriction on a basic role Q, denoted $\exists Q$. A *(general) concept* C is either a basic concept B or the negation of a basic concept $\neg B$.

Statements about roles and concepts are expressed via axioms, where an *axiom* is either (1) a *concept inclusion axiom* $B \sqsubseteq C$, where B is a basic concept, and C is a concept, or (2) a *functionality axiom* $(\mathsf{funct}\ Q)$, where Q is a basic role, or (3) a *concept membership axiom* $A(a)$, where $A \in \mathbf{A}$ and $a \in \mathbf{I}$, or (4) a *role membership axiom* $P(a, c)$, where $P \in \mathbf{R}_A$ and $a, c \in \mathbf{I}$. A *TBox* is a finite set of concept inclusion and functionality axioms. An *ABox* is a finite set of concept and role membership axioms. A *knowledge base* $KB = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. *CQs* and *BCQs* are defined as usual, with concept and role membership axioms as atoms (over variables and individuals as arguments).

The description logic *DL-Lite*$_{\mathcal{R}}$ allows for (5) *role inclusion axioms* $Q \sqsubseteq R$, rather than functionality axioms, where Q is a basic role, and R is a role.

Example 17. Consider the sets of atomic concepts, abstract roles, and individuals $\mathbf{A}$, $\mathbf{R}_A$, and $\mathbf{I}$, respectively, given as follows:

$$\begin{aligned}\mathbf{A} &= \{\textit{Scientist}, \textit{Article}, \textit{ConferencePaper}, \textit{JournalPaper}\},\\ \mathbf{R}_A &= \{\textit{hasAuthor}, \textit{hasFirstAuthor}, \textit{isAuthorOf}\},\\ \mathbf{I} &= \{i_1, i_2\}.\end{aligned}$$

The following concept inclusion axioms express that (i) conference and journal papers are articles, (ii) conference papers are not journal papers, (iii) every scientist has a publication, and (iv) *isAuthorOf* relates scientists and articles:

(i) $\mathit{ConferencePaper} \sqsubseteq \mathit{Article}$, $\mathit{JournalPaper} \sqsubseteq \mathit{Article}$,
(ii) $\mathit{ConferencePaper} \sqsubseteq \neg \mathit{JournalPaper}$, $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthorOf}$,
(iii) $\exists \mathit{isAuthorOf} \sqsubseteq \mathit{Scientist}$, $\exists \mathit{isAuthorOf}^- \sqsubseteq \mathit{Article}$.

Some role inclusion and functionality axioms are as follows; they express that (v) *isAuthorOf* is the inverse of *hasAuthor*, and (vi) *hasFirstAuthor* is functional:

(v) $\mathit{isAuthorOf}^- \sqsubseteq \mathit{hasAuthor}$, $\mathit{hasAuthor}^- \sqsubseteq \mathit{isAuthorOf}$,
(vi) $(\mathsf{funct}\ \mathit{hasFirstAuthor})$.

The following are some concept and role memberships, which express that the individual i_1 is a scientist who authors the article i_2:

$$\mathit{Scientist}(i_1),\ \mathit{isAuthorOf}(i_1, i_2),\ \mathit{Article}(i_2).$$

7.2. Semantics of DL-Lite$_\mathcal{F}$ and DL-Lite$_\mathcal{R}$

The semantics is defined via standard first-order interpretations. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a nonempty (*abstract*) *domain* $\Delta^\mathcal{I}$ and a mapping $\cdot^\mathcal{I}$ that assigns to each atomic concept $C \in \mathbf{A}$ a subset of $\Delta^\mathcal{I}$, to each abstract role $R \in \mathbf{R}_A$ a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and to each individual $a \in \mathbf{I}$ an element of $\Delta^\mathcal{I}$. Here, different individuals are associated with different elements of $\Delta^\mathcal{I}$ (*unique name assumption*). The mapping $\cdot^\mathcal{I}$ is extended to all concepts and roles by:

- $(P^-)^\mathcal{I} = \{(a, b) \mid (b, a) \in P^\mathcal{I}\}$;
- $(\neg Q)^\mathcal{I} = \Delta^\mathcal{I} \times \Delta^\mathcal{I} - Q^\mathcal{I}$;
- $(\exists Q)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \exists y\colon (x, y) \in Q^\mathcal{I}\}$;
- $(\neg B)^\mathcal{I} = \Delta^\mathcal{I} \setminus B^\mathcal{I}$.

The *satisfaction* of an axiom F in the interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models F$, is defined as follows: (1) $\mathcal{I} \models B \sqsubseteq C$ iff $B^\mathcal{I} \subseteq C^\mathcal{I}$; (2) $\mathcal{I} \models (\mathsf{funct}\ Q)$ iff $(o, o') \in Q^\mathcal{I}$ and $(o, o'') \in Q^\mathcal{I}$ implies $o' = o''$; (3) $\mathcal{I} \models A(a)$ iff $a^\mathcal{I} \in A^\mathcal{I}$; (4) $\mathcal{I} \models P(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}$; and (5) $\mathcal{I} \models Q \sqsubseteq R$ iff $Q^\mathcal{I} \subseteq R^\mathcal{I}$. The interpretation $\mathcal{I}$ *satisfies* the axiom F, or $\mathcal{I}$ is a *model* of F, iff $\mathcal{I} \models F$. The interpretation $\mathcal{I}$ *satisfies* a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, or $\mathcal{I}$ is a *model* of KB, denoted $\mathcal{I} \models KB$, iff $\mathcal{I} \models F$ for all $F \in \mathcal{T} \cup \mathcal{A}$. We say that KB is *satisfiable* (resp., *unsatisfiable*) iff KB has a (resp., no) model. The semantics of *CQs* and *BCQs* is as usual in first-order logic.

7.3. Translation of DL-Lite$_\mathcal{F}$ and DL-Lite$_\mathcal{R}$ into Datalog$_0^\pm$

The translation τ from the elementary ingredients and axioms of *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$ into Datalog$_0^\pm$ is defined as follows:

(1) Every atomic concept $A \in \mathbf{A}$ is associated with a unary predicate $\tau(A) = p_A \in \mathcal{R}$, every abstract role $P \in \mathbf{R}_A$ is associated with a binary predicate $\tau(P) = p_P \in \mathcal{R}$, and every individual $i \in \mathbf{I}$ is associated with a constant $\tau(i) = c_i \in \Delta$.

(2) Every concept inclusion axiom $B \sqsubseteq C$ is translated to the TGD or constraint $\tau(B \sqsubseteq C) = \tau'(B) \rightarrow \tau''(C)$, where

  (i) $\tau'(B)$ is defined as $p_A(X)$, $p_P(X, Y)$, and $p_P(Y, X)$, if B is of the form A, $\exists P$, and $\exists P^-$, respectively, and

  (ii) $\tau''(C)$ is defined as $p_A(X)$, $\exists Z\, p_P(X, Z)$, $\exists Z\, p_P(Z,\ X)$, $\neg p_A(X)$, $\neg p_P(X, Y')$, and $\neg p_P(Y', X)$, if C is of form A, $\exists P$, $\exists P^-$, $\neg A$, $\neg \exists P$, and $\neg \exists P^-$, respectively.

(3) The functionality axioms $(\mathsf{funct}\ P)$ and $(\mathsf{funct}\ P^-)$ are under τ translated to the EGDs $p_P(X, Y) \wedge p_P(X, Y') \rightarrow Y = Y'$ and $p_P(X, Y) \wedge p_P(X', Y) \rightarrow X = X'$, respectively.

(4) Every concept membership axiom $A(a)$ is under τ translated to the database atom $p_A(c_a)$, and every role membership axiom $P(a, b)$ to the database atom $p_P(c_a, c_b)$.

(5) Every role inclusion axiom $Q \sqsubseteq R$ is translated to the TGD or constraint $\tau(Q \sqsubseteq R) = \tau'(Q) \rightarrow \tau''(R)$, where

  (i) $\tau'(Q)$ is defined as $p_P(X, Y)$ and $p_P(Y, X)$, if Q is of the form P and P^-, respectively, and

  (ii) $\tau''(R)$ is defined as $p_P(X, Y)$, $p_P(Y, X)$, $\neg p_P(X, Y)$, and $\neg p_P(Y, X)$, if R is of the form P, P^-, $\neg P$, and $\neg P^-$, respectively.

Example 18. The concept inclusion axioms of Example 17 are translated to the following TGDs and constraints (where we identify atomic concepts and roles

with their predicates):

$$\begin{aligned}
&\textit{ConferencePaper}(X) \rightarrow \textit{Article}(X),\\
&\textit{JournalPaper}(X) \rightarrow \textit{Article}(X),\\
&\textit{ConferencePaper}(X) \rightarrow \neg\textit{JournalPaper}(X),\\
&\textit{Scientist}(X) \rightarrow \exists Z\ \textit{isAuthorOf}(X, Z),\\
&\textit{isAuthorOf}(X, Y) \rightarrow \textit{Scientist}(X),\\
&\textit{isAuthorOf}(Y, X) \rightarrow \textit{Article}(X).
\end{aligned}$$

The role inclusion and functionality axioms of Example 17 are translated to the following TGDs and EGDs:

$$\begin{aligned}
&\textit{isAuthorOf}(Y, X) \rightarrow \textit{hasAuthor}(X, Y),\\
&\textit{hasAuthor}(Y, X) \rightarrow \textit{isAuthorOf}(X, Y),\\
&\textit{hasFirstAuthor}(X, Y), \textit{hasFirstAuthor}(X, Y') \rightarrow Y = Y'.
\end{aligned}$$

The concept and role membership axioms of Example 17 are translated to the following database atoms (where we also identify individuals with their constants):

$$\textit{Scientist}(i_1),\ \textit{isAuthorOf}(i_1, i_2),\ \textit{Article}(i_2).$$

Every knowledge base KB in *DL-Lite*$_{\mathcal{S}}$, $\mathcal{S} \in \{\mathcal{F}, \mathcal{R}\}$, is then translated into a database D_{KB}, set of TGDs Σ_{KB}, and disjunction of queries $\mathcal{Q}_{KB}$ as follows: (i) the database D_{KB} is the set of all $\tau(\phi)$ such that ϕ is a concept or role membership axiom in KB, (ii) the set of TGDs Σ_{KB} is the set of all TGDs resulting from $\tau(\phi)$ such that ϕ is a concept or role inclusion axiom in KB, and (iii) $\mathcal{Q}_{KB}$ is the disjunction of all queries resulting from constraints and EGDs $\tau(\phi)$ such that ϕ is a concept inclusion, or role inclusion, or functionality axiom in KB (satisfying any query Q occurring in $\mathcal{Q}_{KB}$ means violating a constraint or EGD).

The following lemma shows that the TGDs generated from a *DL-Lite*$_{\mathcal{R}}$ knowledge base are in fact linear TGDs, and that the TGDs and EGDs generated from a *DL-Lite*$_{\mathcal{F}}$ knowledge base are in fact linear TGDs and NC keys, respectively. Here, the fact that the generated TGDs and EGDs are linear and keys, respectively, is immediate by the above translation. Proving the NC property for the generated keys boils down to showing that keys resulting from functionality axioms $(\mathsf{funct}\ P)$ are NC with TGDs from concept inclusion axioms $B \sqsubseteq \exists P$ and $B \sqsubseteq \exists P^-$.

Lemma 16. *Let KB be a knowledge base in DL-Lite$_{\mathcal{S}}$, $\mathcal{S} \in \{\mathcal{F}, \mathcal{R}\}$. Then, (a) every TGD in Σ_{KB} is linear. If KB is in DL-Lite$_{\mathcal{F}}$, and Σ_K is the set of all EGDs encoded in $\mathcal{Q}_{KB}$, then (b) every EGD in Σ_K is a key, and (c) Σ_K is NC with Σ_{KB}.*

The next result shows that BCQs addressed to knowledge bases in *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$ can be reduced to BCQs in linear Datalog$_0^{\pm}$. This important result follows from the above Lemma 16 and Theorem 14 (stating that the NC property for keys implies their separability relative to a set of TGDs).

Theorem 17. *Let KB be a knowledge base in DL-Lite$_{\mathcal{S}}$, $\mathcal{S} \in \{\mathcal{F}, \mathcal{R}\}$, and let Q be a BCQ for KB. Then, Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$.*

As an immediate consequence, the satisfiability of knowledge bases in *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$ can be reduced to BCQs in Datalog$_0^{\pm}$. Intuitively, the theorem follows from the observation that the unsatisfiability of KB is equivalent to the truth of $\bot$ in KB, which is in turn equivalent to $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.

Theorem 18. *Let KB be a knowledge base in DL-Lite$_{\mathcal{S}}$, $\mathcal{S} \in \{\mathcal{F}, \mathcal{R}\}$. Then, KB is unsatisfiable iff $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.*

The next important result shows that Datalog$_0^{\pm}$ is strictly more expressive than both *DL-Lite*$_{\mathcal{F}}$ and *DL-Lite*$_{\mathcal{R}}$. The main idea behind its proof is to show that neither *DL-Lite*$_{\mathcal{F}}$ nor *DL-Lite*$_{\mathcal{R}}$ can express the TGD $p(X) \rightarrow q(X, X)$.

Theorem 19. *Datalog$_0^{\pm}$ is strictly more expressive than DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$.*

8. Ontology Querying in DL-Lite$_{\mathcal{A}}$

We now generalize the results of Section 7 to the description logic *DL-Lite*$_{\mathcal{A}}$ of the *DL-Lite* family [72]. We first recall the syntax and the semantics of *DL-Lite*$_{\mathcal{A}}$. We then define the translation and the representation and expressivity results.

8.1. Syntax of DL-Lite$_{\mathcal{A}}$

As for the elementary ingredients of *DL-Lite*$_{\mathcal{A}}$, let $\mathbf{D}$ be a finite set of *atomic datatypes* d, which are associated with pairwise disjoint sets of *data values* $\mathbf{V}_d$. Let $\mathbf{A}$, $\mathbf{R}_A$, $\mathbf{R}_D$, and $\mathbf{I}$ be pairwise disjoint sets of *atomic concepts*, *atomic roles*, *atomic attributes*, and *individuals*, respectively, and let $\mathbf{V} = \bigcup_{d \in \mathbf{D}} \mathbf{V}_d$.

Roles, concepts, attributes, and datatypes are defined as follows:

- A *basic role* Q is either an atomic role $P \in \mathbf{R}_A$ or its inverse P^-. A *(general) role* R is either a basic role Q or the negation of a basic role $\neg Q$.

- A *basic concept* B is either an atomic concept $A \in \mathbf{A}$, or an existential restriction on a basic role Q, denoted $\exists Q$, or the domain of an atomic attribute $U \in \mathbf{R}_D$, denoted $\delta(U)$. A *(general) concept* C is either the *universal concept* $\top_C$, or a basic concept B, or the negation of a basic concept $\neg B$, or an existential restriction on a basic role Q of form $\exists Q.C$, where C is a concept.
- A *(general) attribute* V is either an atomic attribute $U \in \mathbf{R}_D$ or the negation of an atomic attribute $\neg U$.
- A *basic datatype* E is the range of an atomic attribute $U \in \mathbf{R}_D$, denoted $\rho(U)$. A *(general) datatype* F is either the *universal datatype* $\top_D$ or an atomic datatype $d \in \mathbf{D}$.

An *axiom* has one of the following forms: (1) $B \sqsubseteq C$ (*concept inclusion axiom*), where B is a basic concept, and C is a concept; (2) $Q \sqsubseteq R$ (*role inclusion axiom*), where Q is a basic role, and R is a role; (3) $U \sqsubseteq V$ (*attribute inclusion axiom*), where U is an atomic attribute, and V is an attribute; (4) $E \sqsubseteq F$ (*datatype inclusion axiom*), where E is a basic datatype, and F is a datatype; (5) $(\mathsf{funct}\ Q)$ (*role functionality axiom*), where Q is a basic role; (6) $(\mathsf{funct}\ U)$ (*attribute functionality axiom*), where U is an atomic attribute; (7) $A(a)$ (*concept membership axiom*), where A is an atomic concept and $a \in \mathbf{I}$; (8) $P(a, b)$ (*role membership axiom*), where P is an atomic role and $a, b \in \mathbf{I}$; and (9) $U(a, v)$ (*attribute membership axiom*), where U is an atomic attribute, $a \in \mathbf{I}$, and $v \in \mathbf{V}$.

We next define knowledge bases, which consist of a restricted finite set of inclusion and functionality axioms, called TBox, and a finite set of membership axioms, called ABox. We first define the restriction on inclusion and functionality axioms. A basic role P or P^- (resp., an atomic attribute U) is an *identifying property* in a set of axioms $\mathcal{S}$ iff $\mathcal{S}$ contains a functionality axiom $(\mathsf{funct}\ P)$ or $(\mathsf{funct}\ P^-)$ (resp., $(\mathsf{funct}\ U)$). Given an inclusion axiom α of the form $X \sqsubseteq Y$ (resp., $X \sqsubseteq \neg Y$), a basic role (resp., atomic attribute) Y appears *positively* (resp., *negatively*) in its right-hand side of α. A basic role (resp., atomic attribute) is *primitive* in $\mathcal{S}$ iff it does not appear positively in the right-hand side of an inclusion axiom in $\mathcal{S}$ and it does not appear in an expression $\exists Q.C$ in $\mathcal{S}$. We can now define knowledge bases. A *TBox* is a finite set $\mathcal{T}$ of inclusion and functionality axioms such that every identifying property in $\mathcal{T}$ is primitive. Intuitively, identifying properties cannot be specialized in $\mathcal{T}$, i.e., they cannot appear positively in the right-hand side of inclusion axioms in $\mathcal{T}$. An *ABox* $\mathcal{A}$ is a finite set of membership axioms. A *knowledge base* $KB = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. As usual, *CQs* and *BCQs* use concept and role membership axioms as atoms (over variables and individuals).

8.2. Semantics of DL-Lite$_\mathcal{A}$

The semantics of *DL-Lite$_\mathcal{A}$* is defined in terms of standard typed first-order interpretations. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of (i) a nonempty domain $\Delta^\mathcal{I} = (\Delta^\mathcal{I}_O, \Delta^\mathcal{I}_V)$, which is the disjoint union of the *domain of objects* $\Delta^\mathcal{I}_O$ and the *domain of values* $\Delta^\mathcal{I}_V = \bigcup_{d \in \mathbf{D}} \Delta^\mathcal{I}_d$, where the $\Delta^\mathcal{I}_d$'s are pairwise disjoint domains of values for the datatypes $d \in \mathbf{D}$, and (ii) a mapping $\cdot^\mathcal{I}$ that assigns to each datatype $d \in \mathbf{D}$ its domain of values $\Delta^\mathcal{I}_d$, to each data value $v \in \mathbf{V}_d$ an element of $\Delta^\mathcal{I}_d$ (such that $v \neq w$ implies $v^\mathcal{I} \neq w^\mathcal{I}$), to each atomic concept $A \in \mathbf{A}$ a subset of $\Delta^\mathcal{I}_O$, to each atomic role $P \in \mathbf{R}_A$ a subset of $\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O$, to each atomic attribute $P \in \mathbf{R}_D$ a subset of $\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V$, to each individual $a \in \mathbf{I}$ an element of $\Delta^\mathcal{I}_O$ (such that $a \neq b$ implies $a^\mathcal{I} \neq b^\mathcal{I}$). Note that different data values (resp., individuals) are associated with different elements of $\Delta^\mathcal{I}_V$ (resp., $\Delta^\mathcal{I}_O$) (*unique name assumption*). The extension of $\cdot^\mathcal{I}$ to all concepts, roles, attributes, and datatypes, and the *satisfaction* of an axiom α in $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models \alpha$, are defined by:

- $(\top_D)^\mathcal{I} = \Delta^\mathcal{I}_V$ and $(\top_C)^\mathcal{I} = \Delta^\mathcal{I}_O$;
- $(\neg U)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V) - U^\mathcal{I}$;
- $(\neg Q)^\mathcal{I} = (\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O) - Q^\mathcal{I}$;
- $(\rho(U))^\mathcal{I} = \{v \in \Delta^\mathcal{I}_V \mid \exists o\colon (o, v) \in U^\mathcal{I}\}$;
- $(\delta(U))^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists v\colon (o, v) \in U^\mathcal{I}\}$;
- $(P^-)^\mathcal{I} = \{(o, o') \in \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O \mid (o', o) \in P^\mathcal{I}\}$;
- $(\exists Q)^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists o'\colon (o, o') \in Q^\mathcal{I}\}$;
- $(\exists Q.C)^\mathcal{I} = \{o \in \Delta^\mathcal{I}_O \mid \exists o'\colon (o, o') \in Q^\mathcal{I}, o' \in C^\mathcal{I}\}$;
- $(\neg B)^\mathcal{I} = \Delta^\mathcal{I}_O \setminus B^\mathcal{I}$.

The *satisfaction* of an axiom α in the interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, denoted $\mathcal{I} \models \alpha$, is defined as follows: (1) $\mathcal{I} \models B \sqsubseteq C$ iff $B^\mathcal{I} \subseteq C^\mathcal{I}$, (2) $\mathcal{I} \models Q \sqsubseteq R$ iff $Q^\mathcal{I} \subseteq R^\mathcal{I}$, (3) $\mathcal{I} \models E \sqsubseteq F$ iff $E^\mathcal{I} \subseteq F^\mathcal{I}$, (4) $\mathcal{I} \models U \sqsubseteq V$ iff $U^\mathcal{I} \subseteq V^\mathcal{I}$, (5) $\mathcal{I} \models (\mathsf{funct}\ Q)$ iff $(o, q), (o, q') \in Q^\mathcal{I}$ implies $q = q'$, (6) $\mathcal{I} \models (\mathsf{funct}\ U)$ iff $(o, v), (o, v') \in U^\mathcal{I}$ implies $v = v'$, (7) $\mathcal{I} \models A(a)$ iff $a^\mathcal{I} \in A^\mathcal{I}$, (8) $\mathcal{I} \models P(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I}$, (9) $\mathcal{I} \models U(a, v)$ iff $(a^\mathcal{I}, v^\mathcal{I}) \in U^\mathcal{I}$. The interpretation $\mathcal{I}$ *satisfies* the axiom α, or $\mathcal{I}$ is a *model* of α, iff $\mathcal{I} \models \alpha$. We say $\mathcal{I}$ *satisfies* a knowledge base $KB = (\mathcal{T}, \mathcal{A})$, or $\mathcal{I}$ is a *model* of KB, denoted $\mathcal{I} \models KB$, iff $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T} \cup \mathcal{A}$. We say KB is *satisfiable* (resp., *unsatisfiable*) iff KB has a (resp., no) model. The semantics of *CQs* and *BCQs* is as usual in first-order logic.

8.3. Translation of DL-Lite$_\mathcal{A}$ into Datalog$_0^\pm$

The translation τ from the elementary ingredients and axioms of *DL-Lite$_\mathcal{A}$* into Datalog$_0^\pm$ is defined as follows:

(1) Every data value v has a constant $\tau(v) = c_v \in \Delta$ such that the $\tau(\mathbf{V}_d)$'s for all datatypes $d \in \mathbf{D}$ are pairwise disjoint. Every datatype $d \in \mathbf{D}$ has under τ a predicate $\tau(d) = p_d$ along with the constraint $p_d(X) \wedge p_{d'}(X) \rightarrow \bot$ for all pairwise distinct $d, d' \in \mathbf{D}$. Every atomic concept $A \in \mathbf{A}$ has a unary predicate $\tau(A) = p_A \in \mathcal{R}$, every abstract role $P \in \mathbf{R}_A$ has a binary predicate $\tau(P) = p_P \in \mathcal{R}$, every attribute $U \in \mathbf{R}_D$ has a binary predicate $\tau(U) = p_U \in \mathcal{R}$, and every individual $i \in \mathbf{I}$ has a constant $\tau(i) = c_i \in \Delta - \bigcup_{d \in \mathbf{D}} \tau(\mathbf{V}_d)$.

(2) Every concept inclusion axiom $B \sqsubseteq C$ is translated to the TGD or constraint $\tau(B \sqsubseteq C) = \tau'(B) \rightarrow \tau''(C)$, where

(i) $\tau'(B)$ is defined as $p_A(X)$, $p_P(X, Y)$, $p_P(Y, X)$, and $p_U(X, Y)$, if B is of the form A, $\exists P$, $\exists P^-$, and $\delta(U)$, respectively, and

(ii) $\tau''(C)$ is defined as $p_A(X)$, $\exists Z\, p_P(X, Z)$, $\exists Z\, p_P(Z, X)$, $\exists Z\, p_U(X, Z)$, $\neg p_A(X)$, $\neg p_P(X, Y')$, $\neg p_P(Y', X)$, $\neg p_U(X, Y')$, $\exists Z\, p_P(X, Z) \wedge p_A(Z)$, and $\exists Z\, p_P(Z, X) \wedge p_A(Z)$, if C is of form A, $\exists P$, $\exists P^-$, $\delta(U)$, $\neg A$, $\neg \exists P$, $\neg \exists P^-$, $\neg \delta(U)$, $\exists P.A$, and $\exists P^-.A$, respectively.

Note that concept inclusion axioms $B \sqsubseteq \top_C$ can be safely ignored, and concept inclusion axioms $B \sqsubseteq \exists Q.C$ can be expressed by the two concept inclusion axioms $B \sqsubseteq \exists Q.A$ and $A \sqsubseteq C$, where A is a fresh atomic concept. Note also that the TGDs with two atoms in their heads abbreviate their equivalent sets of TGDs with singleton atoms in the heads.

(3) The functionality axioms $(\mathsf{funct}\ P)$ and $(\mathsf{funct}\ P^-)$ are under τ translated to the EGDs $p_P(X, Y) \wedge p_P(X, Y') \rightarrow Y = Y'$ and $p_P(X, Y) \wedge p_P(X', Y) \rightarrow X = X'$, respectively. The functionality axiom $(\mathsf{funct}\ U)$ is under τ translated to the EGD $p_U(X, Y) \wedge p_U(X, Y') \rightarrow Y = Y'$.

(4) Every concept membership axiom $A(a)$ is under τ translated to the database atom $p_A(c_a)$. Every role membership axiom $P(a, b)$ is under τ translated to the database atom $p_P(c_a, c_b)$. Every attribute membership axiom $U(a, v)$ is under τ translated to the database atom $p_U(c_a, c_v)$.

(5) Every role inclusion axiom $Q \sqsubseteq R$ is translated to the TGD or constraint $\tau(Q \sqsubseteq R) = \tau'(Q) \rightarrow \tau''(R)$, where

(i) $\tau'(Q)$ is defined as $p_P(X, Y)$ and $p_P(Y, X)$, if Q is of the form P and P^-, respectively, and

(ii) $\tau''(R)$ is defined as $p_P(X,Y)$, $p_P(Y,X)$, $\neg p_P(X,Y)$, and $\neg p_P(Y,X)$, if R is of the form P, P^-, $\neg P$, and $\neg P^-$, respectively.

(6) Attribute inclusion axioms $U \sqsubseteq U'$ and $U \sqsubseteq \neg U'$ are under τ translated to the TGD $p_U(X,Y) \rightarrow p_{U'}(X,Y)$ and the constraint $p_U(X,Y) \rightarrow \neg p_{U'}(X,Y)$, respectively.

(7) Every datatype inclusion axiom $\rho(U) \sqsubseteq d$ is under τ translated to the TGD $p_U(Y,X) \rightarrow p_d(X)$. Note that datatype inclusion axioms $\rho(U) \sqsubseteq \top_D$ can be safely ignored.

Every knowledge base KB in *DL-Lite*$_\mathcal{A}$ is then translated into a database D_{KB}, set of TGDs Σ_{KB}, and disjunction of queries $\mathcal{Q}_{KB}$ as follows: (i) D_{KB} is the set of all $\tau(\phi)$ such that ϕ is a membership axiom in KB along with "type declarations" $p_d(v)$ for all their data values; (ii) Σ_{KB} is the set of all TGDs resulting from $\tau(\phi)$ such that ϕ is an inclusion axiom in KB; and (iii) $\mathcal{Q}_{KB}$ is the disjunction of all queries resulting from datatype constraints and from constraints and EGDs $\tau(\phi)$ such that ϕ is an inclusion or functionality axiom in KB.

The following result shows that Lemma 16 carries over to *DL-Lite*$_\mathcal{A}$. That is, the TGDs and EGDs generated from a *DL-Lite*$_\mathcal{A}$ knowledge base are in fact linear TGDs and NC keys, respectively. This follows from the observation that the new TGDs for *DL-Lite*$_\mathcal{A}$ are also linear or equivalent to collections of linear TGDs, and that the keys are also NC with the new TGDs, due to the restricting assumption that all identifying properties in *DL-Lite*$_\mathcal{A}$ knowledge bases are primitive.

Lemma 20. *Let KB be a knowledge base in DL-Lite$_\mathcal{A}$, and let Σ_K be the set of all EGDs encoded in $\mathcal{Q}_{KB}$. Then, (a) every TGD in Σ_{KB} is linear, (b) every EGD in Σ_K is a key, and (c) Σ_K is NC with Σ_{KB}.*

Consequently, also Theorem 17 carries over to *DL-Lite*$_\mathcal{A}$. That is, BCQs addressed to knowledge bases in *DL-Lite*$_\mathcal{A}$ can be reduced to BCQs in Datalog$_0^\pm$. Note that here and in the theorem below, we assume that every datatype has an infinite number of data values that do not occur in KB.

Theorem 21. *Let KB be a knowledge base in DL-Lite$_\mathcal{A}$, and let Q be a BCQ for KB. Then, Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$.*

Similarly, the satisfiability of knowledge bases in *DL-Lite*$_\mathcal{A}$ can be reduced to BCQs in Datalog$_0^\pm$. This result is formally expressed by the following theorem, which is an extension of Theorem 18 to *DL-Lite*$_\mathcal{A}$.

Theorem 22. *Let KB be a knowledge base in DL-Lite$_\mathcal{A}$. Then, KB is unsatisfiable iff $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.*

Finally, Datalog$_0^\pm$ is also strictly more expressive than *DL-Lite*$_\mathcal{A}$, which is formulated by the next theorem, extending Theorem 19 to *DL-Lite*$_\mathcal{A}$.

Theorem 23. *Datalog$_0^\pm$ is strictly more expressive than DL-Lite$_\mathcal{A}$.*

9. Ontology Querying in Other Description Logics

In this section, we show that also the other tractable description logics of the *DL-Lite* family can be reduced to Datalog$_0^\pm$. We also recall that *F-Logic Lite* is a special case of weakly-guarded Datalog$^\pm$. Furthermore, we show that the tractable description logic $\mathcal{EL}$ can be reduced to guarded Datalog$^\pm$.

9.1. Ontology Querying in the DL-Lite Family

A complete picture of the *DL-Lite* family of description logics (with binary roles) [31] is given in Fig. 4; note that the arrows with filled lines represent proper generalizations, while the ones with dashed lines encode generalizations along with syntactical restrictions. In addition to *DL-Lite*$_\mathcal{F}$, *DL-Lite*$_\mathcal{R}$, and *DL-Lite*$_\mathcal{A}$, the *DL-Lite* family consists of further description logics, namely, (i) *DL-Lite*$_{core}$, which is the intersection of *DL-Lite*$_\mathcal{F}$ and *DL-Lite*$_\mathcal{R}$, (ii) *DL-Lite*$_\mathcal{A}^+$, which is obtained from *DL-Lite*$_\mathcal{A}$ by adding role attributes and identification constraints, and (iii) *DL-Lite*$_{\mathcal{F},\sqcap}$, *DL-Lite*$_{\mathcal{R},\sqcap}$, and *DL-Lite*$_{\mathcal{A},\sqcap}^+$, which are obtained from *DL-Lite*$_\mathcal{F}$, *DL-Lite*$_\mathcal{R}$, and *DL-Lite*$_\mathcal{A}^+$, respectively, by additionally allowing conjunctions in the left-hand sides of inclusion axioms. Furthermore, each above description logic (with binary roles) *DL-Lite*$_X$ has a variant, denoted *DLR-Lite*$_X$, which additionally allows for n-ary relations, along with suitable constructs to deal with them.

Clearly, since *DL-Lite*$_{core}$ is a restriction of *DL-Lite*$_\mathcal{F}$, it can also be reduced to Datalog$_0^\pm$. In the following, we show that all the other description logics of the *DL-Lite* family can similarly be reduced to Datalog$_0^\pm$.

A *role attribute* U_R [29] denotes a binary relation between objects and values. Role attributes come along with: (1) introducing a set of atomic role attributes, (2) extending (general) concepts by expressions of the form $\delta_F(U)$, $\exists\delta_F(U_R)$, and $\exists\delta_F(U_R)^-$, denoting the set of all objects, object pairs, and inverses of object pairs, respectively, that an atomic (concept) attribute U or an atomic role attribute U_R relates to values of a (general) datatype F, (3) extending basic datatypes by atomic datatypes d and the expression $\rho(U_R)$, denoting the range of the atomic

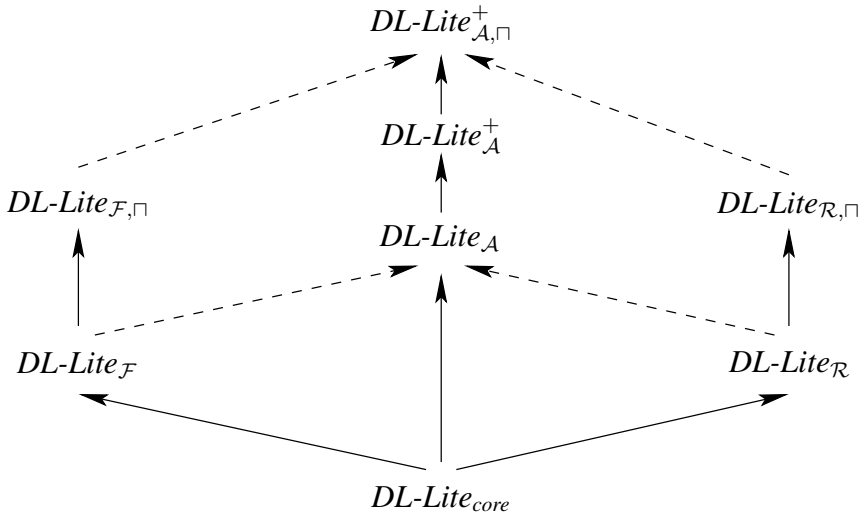

Figure 4: The *DL-Lite* family of description logics.

role attribute U_R, (4) extending (general) datatypes by basic datatypes E and their negations $\neg E$, (5) adding (general) role attributes V_R, which are either atomic role attributes U_R or their negations $\neg U_R$, (6) extending basic roles by the expressions $\delta(U_R)$ and $\delta(U_R)^-$, denoting the set of all object pairs and inverses of object pairs, respectively, that an atomic role attribute U_R relates to values, (7) extending (general) roles by the expressions $\delta_F(U_R)$ and $\delta_F(U_R)^-$, denoting the set of all object pairs and inverses of object pairs, respectively, that an atomic role attribute U_R relates to values of a general datatype F, (8) adding role attribute inclusion axioms $U_R \sqsubseteq V_R$ and functionality axioms $(\mathsf{funct}\ U_R)$, where U_R (resp., V_R) is an atomic (resp., a general) role attribute, (9) adding membership axioms $U_R(a, b, c)$ for atomic role attributes U_R, and (10) extending the notion of identifying property to also include all atomic role attributes in functionality axioms. Note that all axioms with concepts $\exists Q.C$ and with concepts and roles containing the operator δ_F can be reduced to other axioms without them [29]. The translation τ of *DL-Lite*$_{\mathcal{A}}$ into Datalog$_0^{\pm}$ is then extended to the remaining axioms by:

(1) for concept inclusion axioms $B \sqsubseteq C$, we additionally define (i) $\tau'(B)$ as $p_{U_R}(X, Y, Y')$ and $p_{U_R}(Y, X, Y')$, if B is of the form $\exists\delta(U_R)$ and $\exists\delta(U_R)^-$, respectively, and (ii) $\tau''(C)$ as $\exists Z, Z'\, p_{U_R}(X, Z, Z')$, $\exists Z, Z'\, p_{U_R}(Z, X, Z')$, $\neg p_{U_R}(X, Z, Z')$, and $\neg p_{U_R}(Z, X, Z')$ if C is of the form $\exists\delta(U_R)$, $\exists\delta(U_R)^-$, $\neg\exists\delta(U_R)$, and $\neg\exists\delta(U_R)^-$, respectively;

(2) functionality axioms (funct U_R) are under τ translated to the EGD $p_{U_R}(X, Y, Z) \wedge p_{U_R}(X, Y, Z') \rightarrow Z = Z'$;

(3) role attribute membership axioms $U_R(a, b, c)$ are under τ translated to the database atom $p_{U_R}(c_a, c_b, c_c)$;

(4) for role inclusion axioms $Q \sqsubseteq R$, we additionally define (i) $\tau'(Q)$ as $p_{U_R}(X, Y, Y')$ and $p_{U_R}(Y, X, Y')$ if Q is of the form $\delta(U_R)$ and $\delta(U_R)^-$, respectively, and (ii) $\tau''(R)$ as $\exists Z\, p_{U_R}(X, Y, Z)$, $\exists Z\, p_{U_R}(Y, X, Z)$, $\neg p_{U_R}(X, Y, Z)$, and $\neg p_{U_R}(Y, X, Z)$ if R is of the form $\delta(U_R)$, $\delta(U_R)^-$, $\neg\delta(U_R)$, and $\neg\delta(U_R)^-$, respectively;

(5) role attribute inclusion axioms $U_R \sqsubseteq U'_R$ and $U_R \sqsubseteq \neg U'_R$ are under τ translated to the TGD $p_{U_R}(X, Y, Z) \rightarrow p_{U'_R}(X, Y, Z)$ and the constraint $p_{U_R}(X, Y, Z) \rightarrow \neg p_{U'_R}(X, Y, Z)$, respectively; and

(6) datatype inclusion axioms $E \sqsubseteq F$ are under τ translated to $\tau'(E) \rightarrow \tau''(F)$, where (i) $\tau'(E)$ is defined as $p_d(X)$ and $p_{U_R}(Y, Y', X)$, if E is of form d and $\rho(U_R)$, respectively, and (ii) $\tau''(F)$ as $p_d(X)$, $p_{U_R}(Z, Z', X)$, $\neg p_d(X)$, and $\neg p_{U_R}(Z, Z', X)$, if F is of form d, $\rho(U_R)$, $\neg d$, and $\neg\rho(U_R)$, respectively.

An *identification axiom* [30] is of the form (id $B\ I_1, \ldots, I_n$), with $n \geqslant 1$, where B is a basic concept, and each I_j, $j \in \{1, \ldots, n\}$, is either an atomic attribute or a basic role. Such an axiom encodes that the combination of properties $I_1, \ldots, I_n$ identifies the instances of the basic concept B. The notion of identifying property is then extended to also include all atomic attributes and basic roles that occur in identification axioms. The translation τ of *DL-Lite*$_\mathcal{A}$ into Datalog$_0^\pm$ is extended by mapping each such axiom under τ to the EGD (which is a slight extension of Datalog$_0^\pm$ to also include $I_1, \ldots, I_n$ as a key of a virtual relation $R(B, I_1, \ldots, I_n)$):

$$\tau_B(X) \wedge \textstyle\bigwedge_{i=1}^n \tau_{I_i}(X, Y_i) \wedge \tau_B(X') \wedge \bigwedge_{i=1}^n \tau_{I_i}(X', Y_i) \rightarrow X = X'\,,$$

where (i) $\tau_B(X)$ is defined as $p_A(X)$, $p_P(X, Y)$, $p_P(Y, X)$, $p_{U_R}(X, Y, Y')$, $p_{U_R}(Y, X, Y')$, and $p_U(X, Y)$, if B is of form A, $\exists P$, $\exists P^-$, $\exists\delta(U_R)$, $\exists\delta(U_R)^-$, and $\delta(U)$, respectively, and (ii) $\tau_I(X, Y)$ is $p_U(X, Y)$, $p_P(X, Y)$, $p_P(Y, X)$, $p_{U_R}(X, Y, Y')$, and $p_{U_R}(Y, X, Y')$, if I is of form U, P, P^-, $\delta(U_R)$, and $\delta(U_R)^-$, respectively.

The following result shows that Theorems 21 and 22 carry over to *DL-Lite*$_\mathcal{A}^+$, i.e., both BCQs addressed to knowledge bases KB in DL-Lite$_\mathcal{A}^+$ as well as the satisfiability of such KB can be reduced to BCQs in Datalog$_0^\pm$. Here and in the following, the database D_{KB}, the set of TGDs Σ_{KB}, and the disjunction of queries

$\mathcal{Q}_{KB}$ are defined as in Sections 7 and 8, except that we now use the corresponding extended translation τ, rather than those of Sections 7 and 8, respectively.

Theorem 24. *Let KB be a knowledge base in DL-Lite$^{+}_{\mathcal{A}}$, and let Q be a BCQ for KB. Then, (a) Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$, and (b) KB is unsatisfiable iff $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.*

The three description logics *DL-Lite*$_{\mathcal{F},\sqcap}$, *DL-Lite*$_{\mathcal{R},\sqcap}$, and *DL-Lite*$^{+}_{\mathcal{A},\sqcap}$ are obtained from *DL-Lite*$_{\mathcal{F}}$, *DL-Lite*$_{\mathcal{R}}$, and *DL-Lite*$^{+}_{\mathcal{A}}$, respectively, by additionally allowing conjunctions in the left-hand sides of inclusion axioms. These DLs can be encoded by Datalog$^{\pm}_{0}$ with multi-linear TGDs. To this end, the translation τ of *DL-Lite*$_{\mathcal{F}}$, *DL-Lite*$_{\mathcal{R}}$, and *DL-Lite*$^{+}_{\mathcal{A}}$ into Datalog$^{\pm}_{0}$ is extended by first mapping the left-hand sides of inclusion axioms to the conjunctions of their previous mappings under τ. Every body atom $p(\mathbf{X}, \mathbf{Y})$ in a TGD σ with variables $\mathbf{Y}$ that do not occur in the head of σ is then replaced by a new body atom $p'(\mathbf{X})$, where p' is a fresh predicate, along with adding the TGD $p(\mathbf{X}, \mathbf{Y}) \rightarrow p'(\mathbf{X})$.

The next result shows that Theorems 17, 18, and 24 carry over to *DL-Lite*$_{\mathcal{F},\sqcap}$, *DL-Lite*$_{\mathcal{R},\sqcap}$, and *DL-Lite*$^{+}_{\mathcal{A},\sqcap}$, i.e., both BCQs addressed to knowledge bases KB in these DLs and the satisfiability of such KB are reducible to BCQs in Datalog$^{\pm}_{0}$.

Theorem 25. *Let KB be a knowledge base in DL-Lite$_{\mathcal{F},\sqcap}$, DL-Lite$_{\mathcal{R},\sqcap}$, or DL-Lite$^{+}_{\mathcal{A},\sqcap}$, and let Q be a BCQ for KB. Then, (a) Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$, and (b) KB is unsatisfiable iff $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.*

For each of the above description logics (with binary roles) *DL-Lite*$_X$, the description logic *DLR-Lite*$_X$ is obtained from *DL-Lite*$_X$ by additionally allowing for n-ary relations, along with suitable constructs to deal with them. More concretely, (1) the construct $\exists P$ in basic concepts is generalized to the construct $\exists i : R$, where R is an n-ary relation and $i \in \{1, \ldots, n\}$, which denotes the projection of R on its i-th component; (2) the construct $\exists Q.C$ in (general) concepts is generalized to the construct $\exists i : R.C_1 \ldots C_n$, where R is an n-ary relation, the C_i's are (general) concepts, and $i \in \{1, \ldots, n\}$, which denotes those objects that participate as i-th component to tuples of R where the j-th component is an instance of C_j, for all $j \in \{1, \ldots, n\}$; (3) one additionally allows for functionality axioms $(\mathsf{funct}\ I : R)$, stating the functionality of the i-th component of R; and (4) one additionally allows for inclusion axioms between projections of relations $R_1[i_1, \ldots, i_k] \sqsubseteq R_2[j_1, \ldots, j_k]$, where R_1 is an n-ary relation, $i_1, \ldots, i_k \in \{1, \ldots, n\}$, $i_p \neq i_q$ if $p \neq q$, R_2 is an m-ary relation, $j_1, \ldots, j_k \in \{1, \ldots, m\}$, and $j_p \neq j_q$

if $p \neq q$. Since the construct $\exists i : R.C_1 \ldots C_n$ can be removed in a similar way as $\exists Q.C$ in the binary case [29], it only remains to define the translation τ into Datalog$_0^{\pm}$ for the following cases:

(1) for concept inclusion axioms $B \sqsubseteq C$, we additionally define (i) $\tau'(B)$ as $p_R(Y_1, \ldots, Y_{i-1}, X, Y_{i+1}, \ldots, Y_n)$, if $B = \exists i : R$, and (ii) $\tau''(C)$ as $\exists Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_n\, p_R(Z_1, \ldots, Z_{i-1}, X, Z_{i+1}, \ldots, Z_n)$ and $\neg p_R(Y'_1, \ldots, Y'_{i-1}, X, Y'_{i+1}, \ldots, Y'_n)$, if B is of the form $\exists i : R$ and $\neg \exists i : R$, respectively;

(2) every functionality axiom $(\mathsf{funct}\ i : R)$ is under τ mapped to the set of all EGDs $p_R(Y_1, \ldots, Y_{i-1}, X, Y_{i+1}, \ldots, Y_n) \wedge p_R(Y'_1, \ldots, Y'_{i-1}, X, Y'_{i+1}, \ldots, Y'_n) \rightarrow Y_j = Y'_j$ such that $j \in \{1, \ldots, n\}$ and $j \neq i$; and

(3) every inclusion axiom $R_1[i_1, \ldots, i_k] \sqsubseteq R_2[j_1, \ldots, j_k]$ is under τ mapped to the TGD $p_{R_1}(\mathbf{X}) \rightarrow \exists \mathbf{Z}\, p_{R_2}(\mathbf{Z}')$, where $Z'_{j_l} = X_{i_l}$ for all $l \in \{1, \ldots, k\}$, and $\mathbf{Z}$ is the vector of all variables Z'_j with $j \in \{1, \ldots, m\} - \{j_1, \ldots, j_k\}$.

The next theorem finally shows that Theorem 25 carries over to *DLR-Lite*$_{\mathcal{F},\sqcap}$, *DLR-Lite*$_{\mathcal{R},\sqcap}$, and *DLR-Lite*$^{+}_{\mathcal{A},\sqcap}$ (and so also to all less expressive n-ary description logics of the *DL-Lite* family), i.e., both BCQs addressed to knowledge bases KB in these DLs and the satisfiability of such KB are reducible to BCQs in Datalog$_0^{\pm}$.

Theorem 26. *Let KB be a knowledge base in DLR-Lite$_{\mathcal{F},\sqcap}$, DLR-Lite$_{\mathcal{R},\sqcap}$, or DLR-Lite$^{+}_{\mathcal{A},\sqcap}$, and let Q be a BCQ for KB. Then, (a) Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$, and (b) KB is unsatisfiable iff $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$.*

Finally, observe also that Datalog$_0^{\pm}$ is strictly more expressive than every description logic of the *DL-Lite* family and its extension with n-ary relations, which follows from the next theorem, generalizing Theorems 19 and 23.

Theorem 27. *Datalog$_0^{\pm}$ is strictly more expressive than DL-Lite$_{\mathcal{F},\sqcap}$, DL-Lite$_{\mathcal{R},\sqcap}$, DL-Lite$^{+}_{\mathcal{A},\sqcap}$, DLR-Lite$_{\mathcal{F},\sqcap}$, DLR-Lite$_{\mathcal{R},\sqcap}$, and DLR-Lite$^{+}_{\mathcal{A},\sqcap}$.*

9.2. Ontology Querying in F-Logic Lite and $\mathcal{EL}$

Other ontology languages that are reducible to Datalog$^{\pm}$ include *F-Logic Lite* [26], which is a special case of weakly-guarded Datalog$^{\pm}$ [20]. We now show that the tractable description logic $\mathcal{EL}$ can be reduced to guarded Datalog$^{\pm}$.

The description logic $\mathcal{EL}$ has the following ingredients. We assume pairwise disjoint sets of *atomic concepts*, *abstract roles*, and *individuals* $\mathbf{A}$, $\mathbf{R}_A$, and $\mathbf{I}$,

respectively. A *concept* is either the *top concept* $\top$, an atomic concept A, an existential role restriction $\exists R.C$, or a conjunction $C \sqcap D$, where R is an abstract role, and C and D are concepts. A *TBox* is a finite set of concept inclusion axioms $C \sqsubseteq D$, where C and D are concepts, while an *ABox* is a finite set of concept and role membership axioms $A(c)$ and $R(c, d)$, respectively, where A is an atomic concept, R is an abstract role, and c and d are individuals. A knowledge base $KB = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

We define a translation τ from $\mathcal{EL}$ to Datalog$^{\pm}$ with guarded TGDs as follows. Atomic concepts, abstract roles, and individuals are translated under τ in the same way as for *DL-Lite*. The same applies to concept and role membership axioms, which produce the database D_{KB} for a given knowledge base KB in $\mathcal{EL}$. As for concept inclusion axioms $C \sqsubseteq D$, we can w.l.o.g. assume that C contains at most one existential role restriction $\exists R.E$, as any other existential role restriction can be replaced by a fresh atomic concept B along with the concept inclusion axiom $\exists R.E \sqsubseteq B$. We then inductively define τ_X, where X is a variable, for all concepts by $\tau_X(\top) = \top$ (i.e., logical truth), $\tau_X(A) = p_A(X)$, $\tau_X(\exists R.C) = \exists Z\,(p_R(X, Z) \wedge \tau_Z(C))$, and $\tau_X(C \sqcap D) = \tau_X(C) \wedge \tau'(\tau_X(D))$, where τ' is a renaming of existentially quantified variables such that $\tau_X(C)$ and $\tau'(\tau_X(D))$ have no such variables in common anymore. We finally define the translation τ for all concept inclusion axioms by $\tau(C \sqsubseteq D) = \tau'_X(C) \rightarrow \tau'_X(D)$, where $\tau'_X(C)$ (resp., $\tau'_X(D)$) is obtained from $\tau_X(C)$ (resp., $\tau_X(D)$) by removing (resp., "moving out") all existential quantifiers. We denote by Σ_{KB} the resulting set of rules for KB. It is not difficult to verify that Σ_{KB} is in fact a finite set of guarded TGDs.

The following immediate result finally shows that the tractable description logic $\mathcal{EL}$ can be reduced to guarded Datalog$^{\pm}$, i.e., BCQs addressed to knowledge bases in $\mathcal{EL}$ can be reduced to BCQs in Datalog$^{\pm}$ with guarded TGDs.

Theorem 28. *Let KB be a knowledge base in $\mathcal{EL}$, and let Q be a BCQ for KB. Then, Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \models Q$.*

10. Stratified Negation

In this section, we extend Datalog$^{\pm}$ by stratified negation. We first define the syntax of TGDs with negations in their bodies (called *normal TGDs*) and of BCQs with negations (called *normal BCQs*), and we introduce a canonical model semantics via iterative chases. We then show that answering safe normal BCQs from databases under stratified sets of guarded normal TGDs can be done on finite portions of these chases; as a consequence, it is data-tractable

(resp., FO-rewritable) in the guarded (resp., linear) case. We finally define a perfect model semantics and show that it coincides with the canonical model semantics, and that it is an isomorphic image of the perfect model semantics of a corresponding normal logic program with function symbols.

We thus provide a natural stratified negation for query answering over ontologies, which has been an open problem in the DL community to date, since it is in general based on several strata of infinite models. By the results of Sections 7 and 8, this also provides a natural stratified negation for the *DL-Lite* family.

10.1. Normal TGDs and BCQs

We first introduce the syntax of normal TGDs, which are informally TGDs that may also contain negated atoms in their bodies. Given a relational schema $\mathcal{R}$, a *normal TGD (NTGD)* has the form $\forall\mathbf{X}\forall\mathbf{Y}\ \Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\Psi(\mathbf{X},\mathbf{Z})$, where $\Phi(\mathbf{X},\mathbf{Y})$ is a conjunction of atoms and negated atoms over $\mathcal{R}$, and $\Psi(\mathbf{X},\mathbf{Z})$ is a conjunction of atoms over $\mathcal{R}$. It is also abbreviated as $\Phi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\Psi(\mathbf{X},\mathbf{Z})$. As in the case of standard TGDs, we can assume that $\Psi(\mathbf{X},\mathbf{Z})$ is a singleton atom. We denote by $head(\sigma)$ the atom in the head of σ, and by $body^+(\sigma)$ and $body^-(\sigma)$ the sets of all positive and negative ("$\neg$"-free) atoms in the body of σ, respectively. We say that σ is *guarded* iff it contains a positive atom in its body that contains all universally quantified variables of σ. We say that σ is *linear* iff σ is guarded and has exactly one positive atom in its body.

As for the semantics of normal TGDs σ, we say that σ is *satisfied* in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h for all the variables and data constants in the body of σ that maps (i) all atoms of $body^+(\sigma)$ to atoms of D and (ii) no atom of $body^-(\sigma)$ to atoms of D, then there exists an extension h' of h that maps all atoms of $head(\sigma)$ to atoms of D.

We next add negation to BCQs as follows. A *normal Boolean conjunctive query (NBCQ)* Q is an existentially closed conjunction of atoms and negated atoms

$$\exists\mathbf{X}\, p_1(\mathbf{X}) \wedge \cdots \wedge p_m(\mathbf{X}) \wedge \neg p_{m+1}(\mathbf{X}) \wedge \cdots \wedge \neg p_{m+n}(\mathbf{X}),$$

where $m \geqslant 1$, $n \geqslant 0$, and the variables of the p_i's are among $\mathbf{X}$. We denote by Q^+ (resp., Q^-) the set of all positive (resp., negative ("$\neg$"-free)) atoms of Q. We say Q is *safe* iff every variable in a negative atom in Q also occurs in a positive atom in Q.

Example 19. Consider the following set of guarded normal TGDs Σ, expressing that (1) if a driver has a non-valid license and drives, then he violates a traffic law, and (2) a license that is not suspended is valid:

$$\sigma\colon hasLic(D,L), drives(D), \neg valid(L) \rightarrow \exists I\ viol(D,I)\,;$$
$$\sigma'\colon hasLic(D,L), \neg susp(L) \rightarrow valid(L)\,.$$

Then, asking whether John commits a traffic violation and whether there exist traffic violations without driving can be expressed by the two safe normal BCQs $Q_1 = \exists X\ viol(john, X)$ and $Q_2 = \exists D, I\ viol(D, I) \wedge \neg drives(D)$, respectively.

10.2. Canonical Model Semantics

We now define the concept of a stratification for normal TGDs as well as the canonical model semantics of databases under stratified sets of guarded normal TGDs via iterative chases along a stratification. We then provide several semantic results around canonical models, and we finally define the semantics of safe normal BCQs via such canonical models.

We define the notion of stratification for normal TGDs by generalizing the classical notion of stratification for Datalog with negation but without existentially quantified variables [6] as follows. A *stratification* of a set of normal TGDs Σ is a mapping $\mu\colon \mathcal{R} \to \{0, 1, \ldots, k\}$ such that for each normal TGD $\sigma \in \Sigma$:

(i) $\mu(pred(head(\sigma))) \geqslant \mu(pred(\mathbf{a}))$ for all $\mathbf{a} \in body^+(\sigma)$;
(ii) $\mu(pred(head(\sigma))) > \mu(pred(\mathbf{a}))$ for all $\mathbf{a} \in body^-(\sigma)$.

We call $k \geqslant 0$ the *length* of μ. For every $i \in \{0, \ldots, k\}$, we then define $D_i = \{\mathbf{a} \in D \mid \mu(pred(\mathbf{a})) = i\}$ and $D_i^\star = \{\mathbf{a} \in D \mid \mu(pred(\mathbf{a})) \leqslant i\}$, as well as $\Sigma_i = \{\sigma \in \Sigma \mid \mu(pred(head(\sigma))) = i\}$ and $\Sigma_i^\star = \{\sigma \in \Sigma \mid \mu(pred(head(\sigma))) \leqslant i\}$. We say that Σ is *stratified* iff it has a stratification μ of some length $k \geqslant 0$.

Example 20. Consider again the set of guarded normal TGDs Σ of Example 19. It is then not difficult to verify that the mapping μ where $\mu(susp) = \mu(hasLic) = \mu(drives) = 0$, $\mu(valid) = 1$, and $\mu(viol) = 2$ is a stratification of Σ of length 2. Hence, Σ is stratified, and we obtain $\Sigma_0 = \emptyset$, $\Sigma_1 = \{\sigma'\}$, and $\Sigma_2 = \{\sigma\}$.

We next define the notion of indefinite grounding, which extends the standard grounding (where rules are replaced by all their possible instances over constants) towards existentially quantified variables. A subset of the set of nulls Δ_N is partitioned into infinite sets of nulls $\Delta_{\sigma,Z}$ (which can be seen as Skolem terms by which Z can be replaced), one for every $\sigma \in \Sigma$ (where Σ is a set of guarded normal TGDs) and every existentially quantified variable Z in σ. An *indefinite instance* of a normal TGD σ is obtained from σ by replacing every universally quantified variable by an element from $\Delta \cup \Delta_N$ and every existentially quantified variable Z by an element from $\Delta_{\sigma,Z}$. The *indefinite grounding* of Σ, denoted $ground(\Sigma)$, is the set of all its indefinite instances. We denote by HB_Σ the set of all atoms built from predicates from Σ and arguments from $\Delta \cup \Delta_N$. We naturally

extend the oblivious chase of D and Σ to databases D with nulls, which are treated as new data constants; similarly, normal TGDs are naturally extended by nulls.

We are now ready to define canonical models of databases under stratified sets of guarded normal TGDs via iterative chases as follows. Note that this is slightly different from [21], where we define the canonical model semantics as iterative universal models. The main reason why we use the slightly stronger definition here is that it makes canonical models isomorphic to canonical models of a corresponding normal logic program with function symbols (cf. Corollary 38).

Definition 5. Let $\mathcal{R}$ be a relational schema. Given a database D for $\mathcal{R}$ under a stratified set of guarded normal TGDs Σ on $\mathcal{R}$, we define the sets S_i along a stratification $\mu\colon \mathcal{R} \to \{0, 1, \ldots, k\}$ of Σ as follows:

(i) $S_0 = chase(D, \Sigma_0)$;
(ii) if $i > 0$, then $S_i = chase(S_{i-1}, \Sigma_i^{S_{i-1}})$, where the set of TGDs $\Sigma_i^{S_{i-1}}$ is obtained from $ground(\Sigma_i)$ by (i) deleting all σ such that $body^-(\sigma) \cap S_{i-1} \neq \emptyset$ and (ii) removing the negative body from the remaining σ's.

Then, S_k is a *canonical model* of D and Σ.

Example 21. Consider again the set of guarded normal TGDs Σ of Example 19 and the database $D = \{susp(l),\ drives(john, c),\ hasLic(john, l)\}$. Since $\Sigma_0 = \emptyset$, $\Sigma_1 = \{\sigma'\}$, and $\Sigma_2 = \{\sigma\}$ (see Example 20), we obtain $S_0 = S_1 = D$, and S_2 is isomorphic to $D \cup \{viol(john, i)\}$, where i is a null.

Observe that, given a database D and a set of guarded TGDs Σ, the oblivious chase of D and Σ minimizes equality between newly introduced nulls, but every TGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$ generates exactly one new null for every indefinite ground instance of the body of σ satisfied in the oblivious chase. The following theorem shows that this policy is actually closely related to the least Herbrand model semantics of positive logic programs with function symbols: there exists an isomorphism from the oblivious chase of D and Σ to the least Herbrand model of a corresponding positive logic programs with function symbols. This provides a strong justification for using the oblivious chase in the above canonical model semantics of databases under stratified sets of guarded normal TGDs.

Given a set of guarded TGDs Σ, the *functional transformation* of Σ, denoted Σ^f, is obtained from Σ by replacing each TGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$ in Σ by the generalized TGD $\sigma^f = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \Psi(\mathbf{X}, \mathbf{f}_\sigma(\mathbf{X}, \mathbf{Y}))$, where $\mathbf{f}_\sigma$ is a vector of function symbols $f_{\sigma,Z}$ for σ, one for every variable Z in $\mathbf{Z}$. Observe that σ^f

now contains function symbols, but no existential quantifiers anymore. Furthermore, for every database D, it holds that $D \cup \Sigma^f$ is a positive logic program with function symbols, which has a canonical semantics via unique least Herbrand models. The notions of databases, queries, and models are naturally extended by such function symbols. An additional restriction on the notion of isomorphism is that nulls $c \in \Delta_{\sigma,Z}$ are associated with terms of the form $f_{\sigma,Z}(\mathbf{x}, \mathbf{y})$.

Theorem 29. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. Then, there exists an isomorphism from $chase(D, \Sigma)$ to the least Herbrand model M of D and Σ^f.*

The following result shows that canonical models of databases D under stratified sets of guarded normal TGDs Σ are in fact also models of D and Σ, which is a minimal property expected from the notion of "canonical model". The proof is done by induction along a stratification of Σ, showing that every S_i is a model of D and $\Sigma_i^\star$, using the chase construction of every S_i. Thus, in particular, the canonical model S_k of D and Σ is a model of D and $\Sigma_k^\star = \Sigma$.

Proposition 30. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a stratified set of guarded normal TGDs on $\mathcal{R}$. Let S be a canonical model of D and Σ. Then, S is also a model of D and Σ.*

In general, there are several canonical models of databases D under stratified sets of guarded normal TGDs Σ. The next result shows that they are all isomorphic. It is proved by induction along a stratification of Σ, showing that for any two constructions of canonical models $S_0, \ldots, S_k$ and $T_0, \ldots, T_k$, it holds that every S_i is isomorphic to T_i, using the chase construction of every S_i and T_i. Thus, in particular, the two canonical models S_k and T_k of D and Σ are isomorphic.

Proposition 31. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a stratified set of guarded normal TGDs on $\mathcal{R}$. Let U and V be two canonical models of D and Σ. Then, U is isomorphic to V.*

We finally define the semantics of safe normal BCQs addressed to databases D under stratified sets of guarded normal TGDs Σ via their canonical models as follows. A BCQ Q *evaluates to true* in D and Σ, denoted $D \cup \Sigma \models_{strat} Q$, iff there exists a homomorphism that maps Q to a canonical model S_k of D and Σ. A safe normal BCQ Q *evaluates to true* in D and Σ, denoted $D \cup \Sigma \models_{strat} Q$, iff there

exists a homomorphism from Q^+ to a canonical model of D and Σ, which cannot be extended to a homomorphism from some $Q^+ \cup \{\mathbf{a}\}$, where $\mathbf{a} \in Q^-$, to the canonical model of D and Σ. Note that the fact that every canonical model of D and Σ is isomorphic to all other canonical models of D and Σ (cf. Proposition 31) assures that the above definition of the semantics of safe normal BCQs does not depend on the chosen canonical model and is thus well-defined.

Example 22. Consider again the set of guarded normal TGDs Σ and the two normal BCQs Q_1 and Q_2 of Example 19. Let the database D be given as in Example 21. Then, by the canonical model S_2 shown in Example 21, Q_1 and Q_2 are answered positively and negatively, respectively.

10.3. Query Answering

As we have seen in the previous section, a canonical model of a database and a stratified set of guarded normal TGDs can be determined via iterative chases, where every chase may be infinite. We next show that for answering safe normal BCQs, it is sufficient to consider only finite parts of these chases. Based on this result, we then show that answering safe normal BCQs in guarded (resp., linear) Datalog$^{\pm}$ with stratified negation is data-tractable (resp., FO-rewritable).

We first give some preliminary definitions as follows. Given a set of atoms S, a database D, and a set of guarded normal TGDs Σ, we denote by $chase^S(D, \Sigma)$ a slightly modified oblivious chase where the TGD chase rule is *applicable* on a guarded normal TGD σ iff the homomorphism h maps every atom in $body^-(\sigma)$ to an atom not from S, and in that case, the TGD chase rule is applied on the TGD obtained from σ by removing the negative body of σ. Then, $g\text{-}chase^{l,S}(D, \Sigma)$ denotes the set of all atoms of depth at most l in the guarded chase forest.

The next result shows that safe normal BCQs Q can be evaluated on finite parts of iterative guarded chase forests of depths depending only on Q and $\mathcal{R}$. Its proof is similar to the proof of Lemma 4. The main difference is that the atoms of Q may now belong to different levels of a stratification, and one also has to check that the negative atoms do not match with any atom in a canonical model.

Theorem 32. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, Σ be a stratified set of guarded normal TGDs on $\mathcal{R}$, and Q be a safe normal BCQ over $\mathcal{R}$. Then, there exists some $l \geqslant 0$, which depends only on Q and $\mathcal{R}$, such that $D \cup \Sigma \models_{strat} Q$ iff Q evaluates to true on S_k, where the sets S_i, $i \in \{0, \ldots, k\}$, are defined as follows:*

(i) $S_0 = g\text{-}chase^l(D, \Sigma_0)$;

(ii) if $i > 0$*, then* $S_i = g\text{-}chase^{l,S_{i-1}}(S_{i-1}, \Sigma_i)$.

The following result shows that answering safe normal BCQs in guarded Datalog$^{\pm}$ with stratified negation is data-tractable. Like in the negation-free case, not only homomorphic images of the query atoms are contained in finite portions of iterative guarded chase forests, but also the whole derivations of these images. That is, the theorem is proved similarly as Theorems 5 and 6; the main difference is that the finite portion of the guarded chase forest is now computed for each level of a stratification, and that we now also have to check that the negative atoms cannot be homomorphically mapped to a canonical model.

Theorem 33. *Let* $\mathcal{R}$ *be a relational schema,* D *be a database for* $\mathcal{R}$*,* Σ *be a stratified set of guarded normal TGDs on* $\mathcal{R}$*, and* Q *be a safe normal BCQ over* $\mathcal{R}$*. Then,* $D \cup \Sigma \models_{strat} Q$ *is decidable in polynomial time in the data complexity.*

The next result shows that answering safe normal BCQs in linear Datalog$^{\pm}$ with stratified negation is FO-rewritable. Its proof extends the line of argumentation in Theorem 9 and Corollary 10 by stratified negation and safe normal BCQs.

Theorem 34. *Let* $\mathcal{R}$ *be a relational schema,* D *be a database for* $\mathcal{R}$*,* Σ *be a stratified set of linear normal TGDs on* $\mathcal{R}$*, and* Q *be a safe normal BCQ over* $\mathcal{R}$*. Then,* Q *is FO-rewritable.*

10.4. Perfect Model Semantics and Independence from Stratification

We now introduce the perfect model semantics of guarded Datalog$^{\pm}$ with stratified negation, and prove that it coincides with the canonical model semantics, and that it is an isomorphic image of the perfect model semantics of the corresponding normal logic program with function symbols. This gives strong evidence that the canonical model semantics of guarded Datalog$^{\pm}$ is quite natural; it also implies that the canonical model semantics is independent of a concrete stratification.

The perfect model semantics of databases D and a stratified set of guarded normal TGDs Σ is defined via a preference relation $\ll$ on the models of D and Σ that are isomorphic images of models of D and Σ^f. We first define the strict and reflexive relations $\prec$ and $\preccurlyeq$ on ground atoms (having terms with function symbols as arguments). Given a set of guarded normal TGDs Σ, the relations $\prec$ and $\preccurlyeq$ on the set of all ground atoms are the smallest relations that satisfy (i) to (iv):

(i) $\mu(head(\sigma)) \preccurlyeq \mu(\mathbf{a})$ for every $\sigma \in ground(\Sigma^f)$ and every $\mathbf{a} \in body^+(\sigma)$,
(ii) $\mu(head(\sigma)) \prec \mu(\mathbf{a})$ for every $\sigma \in ground(\Sigma^f)$ and every $\mathbf{a} \in body^-(\sigma)$,

(iii) $\prec$ and $\preccurlyeq$ are transitively closed, and
(iv) $\prec$ is a subset of $\preccurlyeq$.

We are now ready to define the preference relation $\ll$ on isomorphic images of models of D and Σ^f, as well as the perfect model semantics of D and Σ as a collection of isomorphic such images that are preferred to all others.

Definition 6. Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded normal TGDs on $\mathcal{R}$. For isomorphic images $M, N \subseteq HB_\Sigma$ of two models M^f and N^f of D and Σ^f, respectively, we say that M is *preferable* to N, denoted $M \ll N$, iff (i) M is not isomorphic to N and (ii) for every $a \in M^f - N^f$, there exists some $b \in N^f - M^f$ such that $a \prec b$ (which is also denoted $M^f \ll N^f$). Given an isomorphic image $M \subseteq HB_\Sigma$ of a model of D and Σ^f, we say that M is a *perfect model* of D and Σ iff $M \ll N$ for all isomorphic images $N \subseteq HB_\Sigma$ of models of D and Σ^f such that N is not isomorphic to M.

The following lemma shows that the preference relation $\ll$ is well-defined.

Lemma 35. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded normal TGDs on $\mathcal{R}$. Let M_1^f, M_2^f, N_1^f, and N_2^f be models of D and Σ^f, let $M \subseteq HB_\Sigma$ (resp., $N \subseteq HB_\Sigma$) be an isomorphic image of M_1^f and M_2^f (resp., N_1^f and N_2^f). Then, $M_1^f \ll N_1^f$ iff $M_2^f \ll N_2^f$.*

The following result shows that in the negation-free case, perfect models of D and Σ are isomorphic to the least model of D and Σ^f. Hence, by Theorem 29, they are also isomorphic to the oblivious chase of D and Σ.

Proposition 36. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded TGDs on $\mathcal{R}$. Then, M is a perfect model of D and Σ iff M is an isomorphic image of the least model of D and Σ^f.*

The next result shows how perfect models of D and Σ can be iteratively constructed. Here, given a stratification $\mu \colon \mathcal{R} \to \{0, 1, \ldots, k\}$ of Σ, we define HB_i (resp., $HB_i^\star$) as the set of all $\mathbf{a} \in HB_\Sigma$ with $\mu(pred(\mathbf{a})) = i$ (resp., $\mu(pred(\mathbf{a})) \leqslant i$).

Proposition 37. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a set of guarded normal TGDs on $\mathcal{R}$ with stratification $\mu \colon \mathcal{R} \to \{0, 1, \ldots, k\}$. Let $S \subseteq HB_i^\star$ and $S' \subseteq HB_{i+1}^\star$ such that $S' \cap HB_i^\star = S$. Then, for all $i \in \{0, 1, \ldots, k-1\}$, S' is a perfect model of $D_{i+1}^\star$ and $\Sigma_{i+1}^\star$ iff (i) S is a perfect model of $D_i^\star$ and $\Sigma_i^\star$, and S is an isomorphic image of a model S^f of $D_i^\star$ and $(\Sigma_i^\star)^f$, and (ii) S' is an isomorphic image of the least model of $S^f \cup D_{i+1}$ and $(\Sigma_{i+1}^f)^{S^f}$.*

Observe that the perfect model of the normal logic program $D \cup \Sigma^f$ is given by M^k, where M^0 is the least model of $D_0 \cup \Sigma_0^f$, and every M^{i+1}, $i \in \{0, 1, \ldots, k-1\}$, is the least model of $M^i \cup D_{i+1} \cup (\Sigma_{i+1}^f)^{M^i}$. Hence, as an immediate corollary of Propositions 36 and 37, we obtain that the perfect model of D and Σ is an isomorphic image of the perfect model of D and Σ^f.

Corollary 38. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a stratified set of guarded normal TGDs on $\mathcal{R}$. Then, M is a perfect model of D and Σ iff M is an isomorphic image of the perfect model of D and Σ^f.*

The following theorem shows that the perfect model semantics coincides with the canonical model semantics. It is proved using Theorem 29 and Propositions 36 and 37. Since perfect models are independent of a concrete stratification, the theorem also implies that the same holds for the canonical model semantics.

Theorem 39. *Let $\mathcal{R}$ be a relational schema, D be a database for $\mathcal{R}$, and Σ be a stratified set of guarded normal TGDs on $\mathcal{R}$. Then, M is a canonical model of D and Σ iff M is a perfect model of D and Σ.*

11. Related Work

In this section, we give a short overview of several approaches to ontology querying. The main motivation behind our research is the Semantic Web. We recall that the vision of the *Semantic Web* [15] has led in the recent years to a new way of conceiving information systems, deeply integrated into the Web and its semantics; in the Semantic Web, the information present on the Web is annotated, so as to be machine-readable; in this way, such information can be integrated and especially queried in information systems, and not merely searched by keywords. This requires a precise sharing of terms by means of an *ontology*, so that the semantics of terms across different sources is clear. Moreover, by using ontologies, it is possible to perform automated reasoning tasks in order to infer new knowledge from the raw information residing at the source on the Web. Underneath the ontology, a data layer represents the raw data present on the Web, in an inherently heterogeneous way. The World Wide Web Consortium (W3C) defines several standards, including the *Resource Description Framework (RDF)* for the data layer; the *Web Ontology Language (OWL)*, based on description logics (DLs); for the ontology layer; the *Rule Interchange Format (RIF)*, currently being defined as a standard, for the rule layer. The RIF does not provide a common

semantics; instead, it aims at offering a common exchange format for rules, given that numerous languages already exist.

We start by discussing the features of DLs employed in Semantic Web reasoning. We review a few works in database theory that are deeply related to Semantic Web reasoning, highlighting the role of the chase formal tool. Datalog is a well-known language for knowledge bases, and we show how its extensions (including the languages presented in this paper) can play a prominent role in ontological querying. We discuss different rewriting techniques, which have been the subject of several relevant researches on ontological querying. We then discuss different approaches to integrate rules and ontologies. After reviewing some ontology reasoning systems, we close by discussing the issue of whether to consider infinite models for the theories constituted by the ontologies.

Description Logics. The ontology layer is highly important in the Semantic Web, and a vast corpus of literature has been produced on it. DLs have been playing a central role in ontology reasoning. DLs are decidable fragments of first-order logic, based on concepts (classes of objects) and roles (binary relations on concepts); several variants of them have been thoroughly investigated, and a central issue is the trade-off between expressive power and computational complexity of the reasoning services. In DL reasoning, a knowledge base usually consists of a TBox (terminological component, i.e., ontology statements on concepts and roles) and an ABox (assertional component, i.e., ontology statements on instances of concepts and roles); the latter corresponds to a data set.

The DL $\mathcal{SHOIQ}$ [48] is one of the most expressive DLs, and it is at the basis of the ongoing standardization of OWL 2, a new version of OWL. Reasoning in $\mathcal{SHOIQ}$ is computationally expensive, and several more tractable languages have been proposed in the Semantic Web community. Among such fragments we mention DLP, $\mathcal{EL}^{++}$, ELP, and the *DL-Lite* family.

DLP [47] is a Horn fragment of OWL, i.e., it is existential-free. Differently from logic programs, which adopt the *closed-world* assumption [73], it adopts the *open-world* assumption (like OWL does), thus being monotonic. The description logic $\mathcal{EL}^{++}$ [9] is an extension of $\mathcal{EL}$ [8, 9] (which allows for conjunction and existential restriction) by nominals, concrete domains, general concept inclusion, and role inclusion; reasoning in $\mathcal{EL}^{++}$ is PTIME-complete. ELP [56] introduces additional features, including role inclusion, local reflexivity, role disjointness, concept product, and qualified role inclusion; reasoning in ELP is PTIME-complete. The *DL-Lite* family [31, 72] of description logics focuses on conjunctive query answering under an instance and a set of *DL-Lite* assertions that con-

stitute the ontology; query answering in *DL-Lite* languages is in AC_0 in the data complexity, due to FO-rewritability of all languages in the *DL-Lite* family. Notice that linear Datalog$^{\pm}$ (with negative constraints and non-conflicting keys) can capture *DL-Lite*, while guarded Datalog$^{\pm}$ can capture $\mathcal{EL}$.

Database Schemata, Ontologies, and Chase. Classic database constraints, as well as more involved ones, can be employed in modeling complex schemata and ontologies. Interestingly, the well-known *inclusion dependencies* [1], common constraints in relational databases, are quite useful in expressing ontologies; for instance, in [22, 24], they are employed together with key dependencies to represent an extension of Entity-Relationship schemata; FO-rewritable subclasses of such schemata are defined by means of graph-based conditions. Notice that the linear TGDs that we use in this paper to capture the *DL-Lite* family are in fact inclusion dependencies. The notion of *chase* [65, 50, 27] of a database against a set of inclusion dependencies is crucial in ontological query answering; such notion has been extended to TGDs [45, 37]. It is important to notice that, in most practical cases in ontological reasoning, the chase does not terminate; the first work to tackle the problem of a non-terminating chase was [50]. In *data exchange*, the chase is necessarily finite; *weakly-acyclic* sets of TGDs is the main class of sets of TGDs that guarantees chase termination [45], later extended by [67, 37]. However, this is not appropriate for ontological databases.

Datalog Extensions. Datalog [1] is a powerful language, but it has some inherent limitations in modeling ontologies, as clearly discussed in [70]. To overcome such limitations, existential quantification was introduced in Datalog (Horn) rules in the form of *value invention* [66, 18]. Datalog rules with value invention are, in fact, TGDs. Interestingly, $\mathcal{EL}$ (which is PTIME-complete; see above) can be expressed by guarded Datalog$^{\pm}$. Indeed, guarded Datalog$^{\pm}$ is more general than $\mathcal{EL}$, as it allows for conjunction of arbitrary predicates in rule bodies (provided that guardedness holds). Local reflexivity can also be easily represented with linear TGDs. Guarded Datalog$^{\pm}$ is extended by *weakly-guarded* Datalog$^{\pm}$ in [20]. The restrictions on the body of guarded TGDs (and of weakly-guarded sets of TGDs [20] as well) cannot express, for instance, the *concept product* [79]; they also cannot capture assertions having a compositions of roles in the body, which are inherently non-guarded. However, a paradigm called *stickiness*, on which *sticky Datalog*$^{\pm}$ and its variants are based, allows for such rules. Sticky sets of TGDs [23, 25] are defined by means of a condition based on variable marking. Like *DL-Lite*, some of the variants of sticky Datalog$^{\pm}$ are FO-rewritable, and therefore tractable. Sticky Datalog$^{\pm}$ also properly extends *DL-Lite*.

The work [43] presents an interesting class of logic programs with function symbols, disjunction, constraints, and negation under the answer set semantics; it is shown that consistency checking and brave reasoning are EXPTIME-complete in this setting, and some lower-complexity fragments are presented.

In [26], the ontology language *F-logic Lite* is presented. It is a limited version of *F-Logic* [51], which is a well-known object-oriented formalism. An *F-logic Lite* schema is represented as a fixed set of TGDs using meta-predicates and a set of ground facts. For a comparison to Datalog$^{\pm}$, we refer to [20].

Recent works concentrate on general semantic characterization of sets of TGDs. The notion of rewriting is strictly connected to that of FINITE UNIFICATION SET (FUS). A FUS is semantically characterized as a set of TGDs that enjoy the following property: for every conjunctive query q, the rewriting q_Σ of q obtained by backward-chaining through unification, according to the rules in Σ, terminates. We refer the reader to [11] for a formal definition. Notice that under certain conditions, as specified in [11], a FUS can be combined with a BOUNDED TREEWIDTH SET (BTS), i.e., a set of TGDs such that the chase under such TGDs has bounded treewidth, while retaining decidability of query answering.

Query Rewriting. Rewritability is a widely used technique for ontology querying. *DL-Lite* languages, as we mentioned above, are FO-rewritable, and the rewriting algorithms for them are based on backward resolution, as others developed for dealing with Entity-Relationship schemata [19] and inclusion dependencies [28]. Such rewriting algorithms produce rewritings as union of conjunctive queries, which are evidently first-order. The work in [71] instead goes beyond first-order rewritability by proposing a Datalog rewriting algorithm for the expressive DL $\mathcal{ELHIO}^{\neg}$, which comprises a limited form of concept and role negation, role inclusion, inverse roles, and nominals, i.e., concept that are interpreted as singletons; conjunctive query answering in $\mathcal{ELHIO}^{\neg}$ is PTIME-complete in data complexity, and the proposed algorithm is also optimal for other ontology languages such as *DL-Lite* – in the case of *DL-Lite*, it produces first-order rewritings instead of Datalog rewritings. Other rewriting techniques for PTIME-complete languages (in data complexity) has been proposed for the description logic $\mathcal{EL}$ [78, 54, 55, 64]. Another approach to rewriting is a combination of rewriting according to the ontology *and* to the data; this was proposed in [52, 53] for *DL-Lite*. This technique achieves better performances in cases where the rewriting according to the ontology alone is very large.

Integration of Rules and Ontologies. The integration of rules with ontologies has recently raised significant interest in the research community, as it allows for com-

bining the high expressive power of rule language with the interoperability provided by ontologies. This is different from the approach of this paper, where we concentrate on the ontology alone (expressed as rules), aiming at attaining the highest expressive power with the least computational cost. Essentially, we can classify the approaches to this integration into two categories: *loose coupling* (or *strict semantic separation*), and *tight coupling* (or *strict semantic integration*).

Loose coupling. In loose coupling, the rules layer consists of a (usually) nonmonotonic language while the ontology layer is expressed in OWL/RDF flavor. The two layers do not have particular restrictions, as their interaction is forced to happen through a "safe interface": rule bodies contain calls to DL predicates, allowing for a mix of closed- and open-world semantics. An example of this approach are dl-programs, together with several extension [41, 42, 39, 40, 62, 63, 82], including probabilistic dl-programs, fuzzy dl-programs, and HEX-programs (which allow for different external sources of knowledge with different semantics). More in detail, HEX programs [41, 42] extend the framework of dl-programs so that they can integrate several sources of knowledge, possibly under different semantics. Probabilistic dl-programs [62] extend dl-programs by probabilistic uncertainty, while, similarly, fuzzy dl-programs [63] consider dl-programs with fuzzy vagueness. A framework for aligning ontologies is added on top of dl-programs in [82]. The work in [83] extends dl-programs to handle priorities. Defeasible reasoning is combined with DLs in [5]; in this work, like in the above cited ones, the DL layer serves merely as an input for the default reasoning system; a similar approach is followed in the TRIPLE rules engine [80].

Tight coupling. In the tight coupling approach, the existing semantics of rule language is adapted directly in the ontology layer. The above cited DLP [47] is an example of this, as well as the undecidable SWRL [49]; [47] showed a mutual reduction between inference in a fragment of the DL $\mathcal{SHOIQ}$ and a subset of Horn programs. Between DLP and SWRL, several other works extend the expressiveness while retaining decidability, dealing with this trade-off in different ways. Among hybrid approaches, in which DL knowledge bases act as input sources, we find the works [38, 59, 74, 75]. The paper [38] combines plain Datalog (without negation or disjunction) with the DL $\mathcal{ALC}$, obtaining a language called $\mathcal{AL}$-log. In $\mathcal{AL}$-log, concepts in an $\mathcal{ALC}$ knowledge base (the *structural* component) enforce constraints in rule bodies of a Datalog program (the *relational* component). Levy and Rousset in [59] present the CARIN framework, which combines the DL $\mathcal{ALCNR}$ with logic programs in a similar fashion, allowing also roles to enforce constraints on rules (unlike [38] which allows only concepts to impose constraints). Such interaction leads to undecidability easily, but in [59] two de-

cidable fragments are singled out. Another work along the same lines is [69]. Rosati's *r-hybrid* knowledge bases [74, 75] combine disjunctive Datalog (with classical and default negation) with $\mathcal{ALC}$ based on a generalized answer set semantics; besides the satisfiability problem, also that of answering ground atomic queries is discussed. This formalism is the basis for a later one, building upon it, called $\mathcal{DL}$ + log [76]. Another approach is found in [68], in the framework of hybrid MKNF knowledge bases, based on the first order variant of Lifschitz's logic MKNF [60]. Other recent approaches to combine rules and ontologies through uniform first-order nonmonotonic formalisms are found in [61, 17].

Systems. Several systems perform reasoning services on ontologies in various flavors. The CEL system [10] is based on $\mathcal{EL}^+$, i.e., $\mathcal{EL}$ with the addition of role inclusion; CEL can perform subsumption in polynomial time, thus aiming at tractable reasoning on large knowledge bases. SNOMED [34] is also based on $\mathcal{EL}$, with the restriction of having acyclic TBoxes only. Snorocket [57] is based on $\mathcal{EL}^{++}$, and achieves good scalability without the restrictions of SNOMED. The research on *DL-Lite* has also given raise to systems, in particular QuOnto [3] and Mastro [31]; both such systems rewrite queries into SQL according to a *DL-Lite* TBox, thus taking advantage of the optimizations of an underlying RDBMS.

Finite Controllability. Finally, we have considered in this paper entailment under arbitrary (finite or infinite) models; when this coincides with entailment under finite models only, it is said that *finite controllability* [50, 77, 12] holds.

12. Conclusion

We have introduced a family of expressive extensions of Datalog, called Datalog$^\pm$, as a new paradigm for query answering and reasoning over ontologies. The Datalog$^\pm$ family admits existentially quantified variables in rule heads, and has suitable restrictions to ensure highly efficient ontology querying. These languages are rather attractive, as they are simple, easy to understand and analyze, decidable, and they have good complexity properties. Furthermore, for ontological query answering and reasoning, they turn out to be extremely versatile and expressive: in fact, guarded Datalog$^\pm$ can express the tractable description logic $\mathcal{EL}$, and languages as simple as linear Datalog$^\pm$ with negative constraints and NC keys (both simple first-order features) can express the whole *DL-Lite* family of tractable description logics (including their generalizations with n-ary relations), which are the most popular tractable ontology languages in the context of the Semantic Web and databases. We have also shown how nonmonotonic stratified negation

(a desirable expressive feature that DLs are currently lacking) can be added to Datalog$^{\pm}$, while keeping ontology querying and reasoning tractable.

Datalog$^{\pm}$ is the first approach to a generalization of database rules and dependencies so that they can express ontological axioms, and it is thus a first step towards closing the gap between the Semantic Web and databases. Datalog$^{\pm}$ paves the way for applying decades of research in databases, e.g., on data integration and data exchange, to the context of the Semantic Web, where there is recently a strong interest on highly scalable formalisms for the Web of Data.

The Datalog$^{\pm}$ family is of interest in its own right; it is still a young research topic, and there are many challenging research problems to be tackled. One interesting topic is to explore how Datalog$^{\pm}$ can be made even more expressive. For example, many DLs allow for restricted forms of transitive closure or constraints. Transitive closure is easily expressible in Datalog, but only through non-guarded rules, whose addition to decidable sets of rules may easily lead to undecidability. Hence, it would be interesting to study under which conditions, closure can be safely added to various versions of Datalog$^{\pm}$. Furthermore, for those logics where query answering is FO-rewritable, the resulting FO-query is usually very large. A topic for future work is to study from a theoretical and a practical point of view how such FO-rewritings can be optimized. Finally, it would also be interesting to explore how other forms of nonmonotonic negation, such as negation under the well-founded and the answer set semantics, can be added to Datalog$^{\pm}$.

Acknowledgments. This work was supported by the EPSRC under the grant EP/ E010865/1 "Schema Mappings and Automated Services for Data Integration and Exchange" and by the European Research Council under the EU's 7th Framework Programme (FP7/2007-2013)/ERC grant 246858 – DIADEM. Georg Gottlob also gratefully acknowledges a Royal Society Wolfson Research Merit Award. Thomas Lukasiewicz was also supported by the German Research Foundation (DFG) under the Heisenberg Programme and by a Yahoo! Research Fellowship.

Appendix A: Proofs for Section 3

Proof of Lemma 1. We give a proof by induction on the number of applications of the TGD chase rule to generate $subtree(\mathbf{a}_1)$ and $subtree(\mathbf{a}_2)$.

Basis: We apply the TGD chase rule to generate a child of $\mathbf{a}_1$ and $\mathbf{a}_2$ in the subtrees. The side atoms in such applications are contained in $type(\mathbf{a}_1)$ and $type(\mathbf{a}_2)$, respectively. Suppose that we are adding an atom $\mathbf{b_1}$ as a child of $\mathbf{a}_1$, applying a TGD $\sigma \in \Sigma$, and using as side atoms $S_1 \subseteq type(\mathbf{a_1})$. Then, there exists another

set $S_2 \subseteq \mathit{type}(\mathbf{a_2})$ that is S-isomorphic to S_1. Hence, we can apply σ to $\mathbf{a}_2$ using S_2 as side atoms, and we obtain an atom $\mathbf{b}_2$ as a child of $\mathbf{a}_2$, which is S-isomorphic to $\mathbf{b}_1$. Thus, we can extend the S-isomorphism between $\mathit{type}(\mathbf{a}_1)$ and $\mathit{type}(\mathbf{a}_2)$ to an S-isomorphism between $\mathit{type}(\mathbf{a}_1) \cup \{\mathbf{b}_1\}$ and $\mathit{type}(\mathbf{a}_2) \cup \{\mathbf{b}_2\}$ by assigning to every fresh null in $\mathbf{b}_1$ the corresponding fresh null in $\mathbf{b}_2$.

Induction: By the induction hypothesis, $\mathit{type}(\mathbf{a}_1) \cup P_1$ and $\mathit{type}(\mathbf{a}_2) \cup P_2$ are S-isomorphic, where every P_i, $i \in \{1, 2\}$, is the set of atoms introduced in $\mathit{subtree}(\mathbf{a}_i)$ during the first k applications of the TGD chase rule. The proof is now analogous to the one of the basis, replacing every $\mathit{type}(\mathbf{a}_i)$, $i \in \{1, 2\}$, by $\mathit{type}(\mathbf{a}_i) \cup P_i$, and considering the $(k + 1)$-th application of the TGD chase rule. □

Proof of Lemma 2. As for the atoms $\mathbf{b}$, at most w arguments from $\mathbf{a}$ and at most w nulls in the two extreme cases may be used as arguments in $\mathbf{b}$. We thus obtain $2w$ possible symbols, which can be placed into at most w argument positions. Hence, the number of all non-$\mathit{dom}(\mathbf{a})$-isomorphic atoms $\mathbf{b}$ is given by $(2w)^w$. As for the types S, since $2w$ possible symbols can be placed into at most w argument positions of $|\mathcal{R}|$ predicates, we obtain at most $(2w)^{w \cdot |\mathcal{R}|}$ different atoms, and $\delta = 2^{(2w)^{w \cdot |\mathcal{R}|}}$ is the number of all its subsets. In summary, the number of all pairs as stated in the theorem is bounded by $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$. □

Proof of Lemma 3. Let $k = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, where w is the maximal arity of a predicate in $\mathcal{R}$. Suppose $\mathit{depth}(\mathbf{b}) > \mathit{depth}(\mathbf{a}) + k$ for one atom $\mathbf{b}$ in the type of $\mathbf{a}$. That is, the path P in the guarded chase forest leading to $\mathbf{b}$ has a length greater than k from the depth of $\mathbf{a}$. Suppose first $\mathbf{b}$ contains no nulls. By Lemma 2, there are two isomorphic atoms $\mathbf{h}$ and $\mathbf{h}'$ (in this order) on P with isomorphic types (see Fig. 5). By Lemma 1, the subtree of $\mathbf{h}$ is isomorphic to the subtree of $\mathbf{h}'$. But then $\mathbf{b}$ is also in the subtree of $\mathbf{h}$, on a path Q that is at least one edge shorter than P, which contradicts the assumption that P is the path leading to $\mathbf{b}$. Suppose next the set of nulls N in $\mathbf{b}$ is nonempty, and consider the common predecessor $\mathbf{c}$ of $\mathbf{a}$ and $\mathbf{b}$ of largest depth. By Lemma 2, there are two N-isomorphic atoms $\mathbf{h}$ and $\mathbf{h}'$ (in this order) on P with N-isomorphic types (see Fig. 5). Since $\mathbf{b}$ is in the type of $\mathbf{a}$, it cannot contain new nulls compared to $\mathbf{a}$. Since $\mathbf{c}$ is the common predecessor of $\mathbf{a}$ and $\mathbf{b}$ of largest depth, $\mathbf{b}$ also cannot contain new nulls compared to $\mathbf{c}$, and thus compared to $\mathbf{h}$ and $\mathbf{h}'$. So, $\mathbf{b}$ is in the types of both $\mathbf{h}$ and $\mathbf{h}'$. By Lemma 1, the subtree of $\mathbf{h}$ is N-isomorphic to the subtree of $\mathbf{h}'$. But then $\mathbf{b}$ is also in the subtree of $\mathbf{h}$, on a path Q that is at least one edge shorter than P, which contradicts the assumption that P is the path leading to $\mathbf{b}$. In summary, all atoms in the type of $\mathbf{a}$ can be obtained on paths of length at most k. □

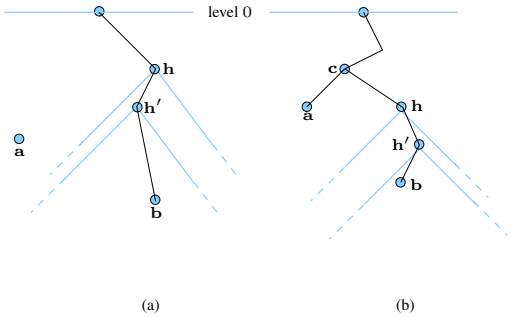

Figure 5: Construction in the proof of Lemma 3.

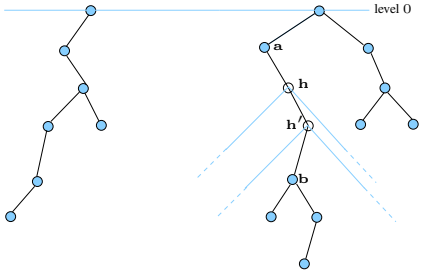

Figure 6: Construction in the proof of Lemma 4.

Proof of Lemma 4. Let $k = n \cdot \delta$, where $n = |Q|$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$. Suppose there exists a homomorphism that maps Q into $chase(D, \Sigma)$. Let μ be a homomorphism of this kind such that $depth(\mu) = \sum_{q \in Q} depth(\mu(q))$ is minimal. We now show that $\mu(Q)$ is contained in $g\text{-}chase^k(D, \Sigma)$. Towards a contradiction, suppose the contrary. Consider the tree consisting of all atoms in $\mu(Q)$ and their ancestors in the guarded chase forest for Σ and D. Since $\mu(Q)$ is not contained in $g\text{-}chase^k(D, \Sigma)$, this tree must contain a path P of length greater than δ of which all inner nodes (i.e., without start and end node) do not belong to $\mu(Q)$ and have no branches (i.e., have exactly one outgoing edge). Let $\mathbf{a}$ be the start node of P. By Lemma 2, there are two $dom(\mathbf{a})$-isomorphic atoms $\mathbf{h}$ and $\mathbf{h}'$ on P with $dom(\mathbf{a})$-isomorphic types. By Lemma 1, $subtree(\mathbf{h})$ is $dom(\mathbf{a})$-isomorphic to $subtree(\mathbf{h}')$. Thus, we can remove $\mathbf{h}$ and the path to $\mathbf{h}'$, obtaining a path that is at least one edge shorter. Let ι be the homomorphism mapping $subtree(\mathbf{h}')$ to $subtree(\mathbf{h})$, and let $\mu' = \mu \circ \iota$. Then, μ' is a homomorphism that maps Q into $chase(D, \Sigma)$ such that $depth(\mu') < depth(\mu)$, which contradicts μ being a homomorphism of this kind such that $depth(\mu)$ is minimal. This shows that $\mu(Q)$ is contained in $g\text{-}chase^k(D, \Sigma)$. $\square$

Proof of Theorem 5. Let Q be a BCQ. Assume there exists a homomorphism μ such that $\mu(Q) \subseteq \mathit{chase}(D, \Sigma)$, which amounts to say that $D \cup \Sigma \models Q$. By Lemma 4, there exists a homomorphism λ such that $\lambda(Q) \subseteq g\text{-}\mathit{chase}^{n \cdot \delta}(D, \Sigma)$, where $n = |Q|$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$. We now show that all ancestors of $\lambda(Q)$ in the chase graph for D and Σ are contained in $g\text{-}\mathit{chase}^{(n+1) \cdot \delta}(D, \Sigma)$. Take any guard $\mathbf{a}$ in $g\text{-}\mathit{chase}^{n \cdot \delta}(D, \Sigma)$. By Lemma 3, for each $\mathbf{b} \in \mathit{type}(\mathbf{a})$, it holds that $\mathit{depth}(\mathbf{b}) \leqslant \mathit{depth}(\mathbf{a}) + \delta$. Hence, with a generic breadth-first application of the TGD chase rule, we do not need to go beyond level $(n+1) \cdot \delta$ to find *all* side atoms of atoms in $g\text{-}\mathit{chase}^{n \cdot \delta}(D, \Sigma)$. $\square$

Proof of Theorem 6. As for membership in P, by the proof of Theorem 5, we obtain the following polynomial decision algorithm. We first construct $g\text{-}\mathit{chase}^{(n+1) \cdot \delta}(D, \Sigma)$, where $n = |Q|$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$, and we then evaluate Q on $g\text{-}\mathit{chase}^{(n+1) \cdot \delta}(D, \Sigma)$.

To prove hardness for P, we give a logspace reduction from the P-complete problem of deciding whether a propositional logic program with at most two body atoms in its rules logically implies a propositional atom [35]. Let L be a propositional logic program with at most two body atoms in its rules, and let p be a propositional atom. So, L is a finite set of rules of the form $h \leftarrow b_1 \wedge b_2$, where h is a propositional atom, and each b_i is either the propositional constant *true*, denoted $\top$, or a propositional atom a_i. Then, we define $\mathcal{R}$, D, Σ, and Q as follows:

$$
\begin{aligned}
\mathcal{R} &= \{\mathit{program},\ \mathit{query},\ \mathit{holds}\};\\
D &= \{\mathit{program}(h, b_1, b_2) \mid h \leftarrow b_1 \wedge b_2 \in L\} \cup \{\mathit{query}(p)\} \cup \{\mathit{holds}(\top)\};\\
\Sigma &= \{\mathit{program}(X, Y, Z) \wedge \mathit{holds}(Y) \wedge \mathit{holds}(Z) \rightarrow \mathit{holds}(X);\\
&\qquad \mathit{query}(X) \wedge \mathit{holds}(X) \rightarrow q\};\\
Q &= q.
\end{aligned}
$$

Observe that only D depends on L and p, while $\mathcal{R}$, Σ, and Q are all fixed. Observe also that D can be computed in logspace from L and p. It is then not difficult to verify that L logically implies p iff $D \cup \Sigma \models Q$. $\square$

Proof of Theorem 7. By the proof of Theorem 5, we can evaluate Q on $g\text{-}\mathit{chase}^{(n+1) \cdot \delta}(D, \Sigma)$, where $n = |Q|$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$, which can be done as follows. For every atom $\mathbf{a} \in D$, we construct the tree of all *potential* descendants in the guarded chase forest of depth up to $(n+1) \cdot \delta$. Since $|\Sigma|$ is constant, every node in this tree has only a constant number of children. Thus, the tree can be constructed in constant time,

and the number of applied instances of TGDs in it is constant. Hence, the union S of all applied instances of TGDs in the trees of descendants of all $\mathbf{a} \in D$ can also be constructed in linear time. Observe now that S is a propositional logic program, and the nodes of the guarded chase forest of depth up to $(n+1) \cdot \delta$ are all atoms that are logically implied by $D \cup S$. Let S' be obtained from S by adding all rules $\mathbf{a} \rightarrow q$ such that (i) $\mathbf{a}$ is a potential node in the guarded chase forest and (ii) Q can be homomorphically mapped to $\mathbf{a}$. Clearly, S' can also be constructed in linear time. Then, $D \cup \Sigma \models Q$ iff $D \cup S' \models q$, where the latter can be decided in linear time [35]. In summary, this shows that deciding $D \cup \Sigma \models Q$ can be done in linear time in the data complexity. $\square$

Appendix B: Proofs for Section 4

Proof of Theorem 9. Let γ_d (which depends only on Q and $\mathcal{R}$) be the depth of the derivation of Q. Let β be the maximum number of body atoms in a TGD in Σ. Then, every atom in the chase is generated by at most β atoms, of which $\beta - 1$ are side atoms. The derivation of $\mathbf{a}$ is therefore contained in all its ancestors; among those, there are at most β^{γ_d} at level 0. If we consider the whole query Q (with $|Q| = n$), the number of level 0-ancestors of its atoms is at most $n \cdot \beta^{\gamma_d}$. An FO-rewriting for Q is thus constructed as follows. Take all possible sets of $n \cdot \beta^{\gamma_d}$ atoms using predicates in $\mathcal{R}$ and having constants from Q and (at most $n \cdot \beta^{\gamma_d} \cdot w$, where w is the maximal arity of a predicate in $\mathcal{R}$) nulls as arguments. Then, considering them as a database B, compute $chase^{\gamma_d}(B, \Sigma)$. Finally, whenever Q can be homomorphically mapped to $chase^{\gamma_d}(B, \Sigma)$, take all atoms in B, transform the nulls into distinct variables, and make the logical conjunction ϕ out of the resulting atoms. The existential closure of the logical disjunction of all such conjunctions ϕ is the rewriting of Q relative to Σ, denoted Q_Σ. Observe now that $D \models Q_\Sigma$ iff $D \cup \Sigma \models Q$ (i.e., $chase(D, \Sigma) \models Q$): this is because every conjunction in Q_Σ corresponds to some derivation of n atoms (soundness), and every derivation of n atoms in the levels of the chase up to γ_d (i.e., all those sufficient to check whether $chase(D, \Sigma) \models Q$) corresponds to a conjunction in Q_Σ (completeness). $\square$

Appendix C: Proofs for Section 6

Proof of Theorem 13. (a) Immediate by the definition of separability.

(b) Assume that Q, Σ_T, and $\mathcal{Q}_C$ can be rewritten into the first-order formula Φ such that for each database D, it holds that $D \cup \Sigma_T \models Q \vee \mathcal{Q}_C$ iff $D \models \Phi$. Now,

let Ψ be the disjunction of all negated EGDs $\neg\sigma$ with $\sigma \in \Sigma_E$. Then, $D \cup \Sigma_T \cup \Sigma_E \models Q \vee \mathcal{Q}_C$ iff $D \models \Phi \vee \Psi$. $\square$

Proof of Theorem 14. The proof in [27] (Lemma 3.8) uses the *restricted* chase (where a TGD does not fire whenever it is satisfied). We construct a somewhat different proof for the oblivious chase, which is used in the present paper. We do this for self-containedness, and also because it is quite instructive to see how the oblivious chase works for TGDs and NC keys. Note that, alternatively, one can just exploit the result shown in [27] (Lemma 3.8) stating that when D satisfies Σ_K, then the restricted chase of D relative to $\Sigma_T \cup \Sigma_K$ is equal to the restricted chase of D relative to Σ_T alone. We observe that the latter is homomorphically equivalent to the oblivious chase $chase(D, \Sigma_T)$. Hence, $chase(D, \Sigma_T \cup \Sigma_K)$ does not fail, and $chase(D, \Sigma_T)$ is a universal model of $D \cup \Sigma_T \cup \Sigma_K$.

We use the standard chase order adopted in this paper: whenever the chase has generated a new atom by some TGD, it applies all applicable keys (EGDs). We show that the oblivious chase converges with such an order. Let $\mathcal{R}$ be a relational schema, Σ_T be a set of TGDs on $\mathcal{R}$, Σ_K be a set of NC keys on $\mathcal{R}$, and D be a database for $\mathcal{R}$. Assume D satisfies Σ_K. By Definition 3, it suffices to show that

(1) when chasing the TGDs in Σ_T over D using the oblivious chase, the keys in Σ_K never lead to a hard violation, and
(2) for every database D and BCQ Q, it holds that $chase(D, \Sigma_T) \models Q$ iff $chase(D, \Sigma_T \cup \Sigma_K) \models Q$.

Suppose that, in the process of constructing the oblivious chase, a TGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, r(\mathbf{X}, \mathbf{Z})$ fires. Take any key $\kappa \in \Sigma_K$ for r, where $\mathbf{K}$ is the corresponding set of positions of relation r. Then, since the set $\mathbf{H}_\sigma$ of positions in $head(\sigma)$ occupied by universally quantified variables is not a proper superset of the set $\mathbf{K}$, there are only two possible cases: (i) $\mathbf{K} = \mathbf{H}_\sigma$: in this case, κ is actually the only key that can fire! By our particular chase order, this key is immediately applied and just eliminates the new atom $\mathbf{a}$ generated by σ, because $\mathbf{a}$ has fresh nulls in all positions but those of $\mathbf{K}$. (ii) At least one position in $\mathbf{K}$ is occupied by an existentially quantified variable in $head(\sigma)$: then, the new fact $\mathbf{a}$ generated by the application of σ contains a fresh null in a position in $\mathbf{K}$, and therefore it cannot violate the key κ. It follows that the oblivious chase only eliminates some facts generated by some TGDs, and converges to a possibly infinite fixpoint Ω without ever producing a hard violation. By results of [37], the resulting $chase(D, \Sigma_T \cup \Sigma_K)$ is a universal model for $D \cup \Sigma_T \cup \Sigma_K$. It is also an endomorphic image of $chase(D, \Sigma_T)$ via the endomorphism $\theta\colon\ chase(D, \Sigma_T) \rightarrow$

$chase(D, \Sigma_T \cup \Sigma_K)$, where θ is defined by the union of all substitutions performed by those keys that are applied. Hence, $chase(D, \Sigma_T)$ and $chase(D, \Sigma_T \cup \Sigma_K)$ are homomorphically equivalent, and thus satisfy the same BCQs. □

Appendix D: Proofs for Section 7

Proof of Lemma 16. Clearly, every TGD generated from KB is linear, which already shows (a). Furthermore, every EGD generated from functionality axioms in KB is obviously a key, which then shows (b). It thus only remains to prove (c). Since every key in Σ_K is defined on a role, the only TGDs that are potentially interacting with Σ_K are those derived from concept inclusion axioms in KB of the form $B \sqsubseteq \exists P$ and $B \sqsubseteq \exists P^-$, when a functionality axiom $(\mathsf{funct}\, P)$ is in KB. In such cases, we have a TGD whose head is of the form $\exists Z\, p_P(X, Z)$ or $\exists Z\, p_P(Z, X)$, and a key of the form $P(Y_3, Y_1), P(Y_3, Y_2) \rightarrow Y_1 = Y_2$. In both cases, (1) the set of key positions $\{p_P[1]\}$ is not a proper subset of the set of X-positions $\{p_P[1]\}$ and $\{p_P[2]\}$, respectively, and (2) the existentially quantified variable Z appears only once in the head of the TGD. That is, the key is non-conflicting with the two TGDs. In summary, Σ_K is non-conflicting with Σ_{KB}. □

Proof of Theorem 17. By Lemma 16, the set Σ_K of all EGDs encoded in $\mathcal{Q}_{KB}$ is a set of keys that is non-conflicting with Σ_{KB}. By Theorem 14, Σ_K is separable from Σ_{KB}. Obviously, Q holds in KB iff $D_{KB} \cup \Sigma_{KB} \cup \Sigma_K \cup \Sigma_C \models Q$, where Σ_C is the set of all constraints encoded in $\mathcal{Q}_{KB}$. As argued in Section 5, the latter is equivalent to $D_{KB} \cup \Sigma_{KB} \cup \Sigma_K \models Q \vee \mathcal{Q}_C$, where $\mathcal{Q}_C$ is the disjunction of all queries resulting from Σ_C. By the definition of separability (cf. Definition 3), the latter is equivalent to $D_{KB} \cup \Sigma_{KB} \models Q \vee \mathcal{Q}_{KB}$. □

Proof of Theorem 18. Observe that KB is unsatisfiable iff the BCQ $Q = \exists X\, A(X)$ holds in $KB' = KB \cup \{A \sqsubseteq B,\ A \sqsubseteq \neg B\}$, where A and B are fresh atomic concepts. By Theorem 17, the latter is equivalent to $D_{KB'} \cup \Sigma_{KB'} \models Q \vee \mathcal{Q}_{KB'}$. This is in turn equivalent to $D_{KB'} \cup \Sigma_{KB'} \models \mathcal{Q}_{KB'}$, that is, $D_{KB} \cup \Sigma_{KB} \models \mathcal{Q}_{KB}$. □

Proof of Theorem 19. The TGD $p(X) \rightarrow q(X, X)$ can neither be expressed in *DL-Lite*$_{\mathcal{F}}$ nor in *DL-Lite*$_{\mathcal{R}}$, since the TGDs of concept and role inclusion axioms can only project away arguments, introduce new nulls as arguments, and change the order of arguments in the predicates for atomic concepts and abstract roles, and the EGDs for functionality axioms can only produce an atom $q(c, c)$ from $q(n, c)$ and/or $q(c, n)$, where n is a null, if $q(c, c)$ was already there before. □

Appendix E: Proofs for Section 8

Proof of Lemma 20. Obviously, every TGD generated from KB is linear or equivalent to a collection of linear TGDs, and every EGD generated from functionality axioms in KB is a key, which already proves (a) and (b). As for (c), we extend the proof of Lemma 16 from *DL-Lite*$_{\mathcal{F}}$ to *DL-Lite*$_{\mathcal{A}}$. As for the TGDs that are potentially interacting with the keys, *DL-Lite*$_{\mathcal{A}}$ newly produces TGDs for role and attribute inclusion axioms $Q \sqsubseteq R$ and $U \sqsubseteq V$, respectively, as well as for concept inclusion axioms $B \sqsubseteq C$, where C may contain general concepts of the form $\exists Q.D$ with basic roles Q and general concepts D. However, by the assumption that all role and attribute functionality axioms can only be expressed on primitive roles and attributes, respectively, the keys are trivially non-conflicting with the new TGDs in the translation from *DL-Lite*$_{\mathcal{A}}$. Therefore, the interesting cases (i.e., those where keys are potentially conflicting with TGDs) are exactly the same as in the proof of Lemma 16, and the rest of the proof goes in the same way. □

Proof of Theorem 21. The proof is verbally nearly identical to the proof of Theorem 17; it only differs in using Lemma 20 instead of Lemma 16. □

Proof of Theorem 22. The proof is verbally nearly identical to the proof of Theorem 18; it only differs in using Theorem 21 instead of Theorem 17. □

Proof of Theorem 23. The proof is verbally nearly identical to the proof of Theorem 19, referring only to *DL-Lite*$_{\mathcal{A}}$ instead of *DL-Lite*$_{\mathcal{F}}$. □

Appendix F: Proofs for Section 9

Proof of Theorem 24. We extend the proofs of Theorems 21 and 22 to *DL-Lite*$^{+}_{\mathcal{A}}$. Observe first that, as for role attributes, the extended translation τ produces linear TGDs and keys. Furthermore, the keys also have the NC property, since (a) by the assumed restriction on *DL-Lite*$^{+}_{\mathcal{A}}$, all the atomic role attributes in functionality axioms $(\mathsf{funct}\ U_R)$ do not occur positively in the right-hand sides of role attribute inclusion axioms, and (b) the key positions resulting from $(\mathsf{funct}\ U_R)$ are not a proper subset of the $\mathbf{X}$-positions in the TGD heads generated from $\delta(U_R)$, $\delta(U_R)^{-}$, $\exists\delta(U_R)$, $\exists\delta(U_R)^{-}$, and $\rho(U_R)$. The extended translation τ for identification axioms $(\mathsf{id}\ B\ I_1, \ldots, I_n)$ lies slightly outside Datalog$^{\pm}_{0}$, since the produced EGD is not really a key, but it can intuitively be considered as a key of the virtual relation $R(B, I_1, \ldots, I_n)$. It also has the NC property, since by the assumed

restriction on *DL-Lite*$^+_{\mathcal{A}}$, all the atomic attributes and basic roles in identification axioms do not occur positively in the right-hand sides of inclusion axioms. □

Proof of Theorem 25. Immediate by Theorems 17, 18, and 24, respectively, as the only difference is that we now have multi-linear TGDs instead of linear ones. □

Proof of Theorem 26. Immediate by Theorem 25, since also the extended translation τ produces only linear TGDs and NC keys. In particular, the NC property of keys follows from the restriction of *DLR-Lite*$_{\mathcal{F},\sqcap}$, *DLR-Lite*$_{\mathcal{R},\sqcap}$, and *DLR-Lite*$^+_{\mathcal{A},\sqcap}$, respectively, that all n-ary relations R in functionality axioms $(\mathsf{funct}\ i : R)$ do not appear positively in the right-hand sides of concept inclusion axioms and of the newly introduced inclusion axioms between projections of relations. □

Proof of Theorem 27. Following the same line of argumentation as in the proof of Theorem 19, it can be shown that the TGD $p(X) \rightarrow q(X, X)$ cannot be expressed in any of the DLs stated in the theorem. □

Appendix G: Proofs for Section 10

Proof of Theorem 29. By induction along the construction of $chase(D, \Sigma)$, we define an isomorphism ι from $chase(D, \Sigma)$ to a subset of M as follows. For every $c \in \Delta \cup \Delta_N$ that occurs in D, we define $\iota(c) = c$. Trivially, ι maps $D \subseteq chase(D, \Sigma)$ isomorphically to $D \subseteq M$. Consider now any step of the construction of $chase(D, \Sigma)$, and let C be the result of the construction thus far. Suppose that ι maps $C \subseteq chase(D, \Sigma)$ isomorphically to a subset of M. Suppose that the next step in the construction of $chase(D, \Sigma)$ is the application of the TGD $\sigma = \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, which produces the atom $\Psi(\mathbf{x}, \mathbf{N})$ from the atoms in $\Phi(\mathbf{x}, \mathbf{y})$. We then extend ι by mapping the vector $\mathbf{N}$ of nulls $N \in \Delta_{\sigma,Z}$, where the Z's are the existentially quantified variables in σ, to the vector $\mathbf{f}_\sigma(\mathbf{x}, \mathbf{y})$ of terms $f_\sigma(\mathbf{x}, \mathbf{y})$. Notice that ι is injective, since every pair $(\mathbf{x}, \mathbf{y})$ uniquely determines the atoms in $\Phi(\mathbf{x}, \mathbf{y})$ and thus at most one application of the TGD σ. Since $\iota(\Phi(\mathbf{x}, \mathbf{y}))$ is a subset of M, and M is a model of D and Σ^f, also $\iota(\Psi(\mathbf{x}, \mathbf{N}))$ must belong to M. Hence, ι also maps the result of applying σ on C to a subset of M. We can thus construct an isomorphism ι from $chase(D, \Sigma)$ to a subset M' of M. But since M' is also a model of D and Σ^f, as otherwise the construction of $chase(D, \Sigma)$ would be incomplete, and since M is the least model of D and Σ^f, we obtain $M' = M$. Thus, ι maps $chase(D, \Sigma)$ isomorphically to M. □

Proof of Proposition 30. Let a stratification of Σ be given by $\mu \colon \mathcal{R} \rightarrow \{0, 1, \ldots, k\}$. By induction on the stratification μ, we now show that for any construction of

a canonical model $S_0, \ldots, S_k$, every S_i is a model of D and $\Sigma_i^\star$. Thus, in particular, the canonical model S_k of D and Σ is a model of D and $\Sigma_k^\star = \Sigma$. Observe first that $(\star)$ if S_i^j, where $j \in \{0, \ldots, i\}$ and $i \in \{0, \ldots, k\}$, is the set of all atoms $\mathbf{a} \in S_i$ such that $\mu(pred(\mathbf{a})) \leqslant j$, then every S_i^j coincides with S_j.

Basis: Since $S_0 = chase(D, \Sigma_0)$, and $chase(D, \Sigma_0)$ is a universal model of D and Σ_0, in particular, S_0 is a model of D and $\Sigma_0 = \Sigma_0^\star$.

Induction: Suppose that S_{i-1} is a model of D and $\Sigma_{i-1}^\star$; we now show that also S_i is a model of D and $\Sigma_i^\star$. Recall first that $S_i = chase(S_{i-1}, \Sigma_i^{S_{i-1}})$. We have to show that (i) D can be homomorphically mapped to S_i and (ii) every $\sigma \in \Sigma_i^\star$ is satisfied in S_i. As for (i), since S_{i-1} is a model of D and $\Sigma_{i-1}^\star$ by the induction hypothesis, D can be homomorphically mapped to S_{i-1}. Since $S_i = chase(S_{i-1}, \Sigma_i^{S_{i-1}})$ is a universal model S_{i-1} and $\Sigma_i^{S_{i-1}}$, it follows that S_{i-1} can be homomorphically mapped to S_i. In summary, D can be homomorphically mapped to S_i. As for (ii), consider any $\sigma \in \Sigma_i^\star$. Then, $\sigma \in \Sigma_j$ for some $j \in \{0, \ldots, i\}$, and since (1) $S_j = chase(S_{j-1}, \Sigma_j^{S_{j-1}})$ is a universal model of S_{j-1} and $\Sigma_j^{S_{j-1}}$ (or $S_j = chase(D, \Sigma_0)$ is a universal model of D and Σ_0, if $j = 0$) and (2) S_j^{j-1} coincides with S_{j-1}, by $(\star)$, it follows that σ is satisfied in S_j, and since S_j coincides with S_i^j, by $(\star)$, it follows that σ is also satisfied in S_i. $\square$

Proof of Proposition 31. Let a stratification of Σ be given by $\mu \colon \mathcal{R} \to \{0, 1, \ldots, k\}$. By induction on the stratification μ, we now show that for any two constructions of canonical models $S_0, \ldots, S_k$ and $T_0, \ldots, T_k$, it holds that S_i is isomorphic to T_i, for every $i \in \{0, \ldots, k\}$. Thus, in particular, the two canonical models S_k and T_k of D and Σ are isomorphic.

Basis: Clearly, $S_0 = chase(D, \Sigma_0)$ and $T_0 = chase(D, \Sigma_0)$ are isomorphic.

Induction: Suppose that S_{i-1} and T_{i-1} are isomorphic; we now show that also S_i and T_i are isomorphic. Since S_{i-1} and T_{i-1} are isomorphic by the induction hypothesis, $\Sigma_i^{S_{i-1}}$ is isomorphic to $\Sigma_i^{T_{i-1}}$. Following the construction of the chase, the isomorphism between S_{i-1} and T_{i-1} can then be extended to an isomorphism between $S_i = chase(S_{i-1}, \Sigma_i^{S_{i-1}})$ and $T_i = chase(T_{i-1}, \Sigma_i^{T_{i-1}})$. $\square$

Proof of Theorem 32. Let $l = n \cdot \delta$, where $n = |Q^+| + 1$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$. The result is proved in the same way as Lemma 4, except that the atoms of Q may now belong to different levels of a stratification (and thus the path P of length greater than δ for the proof by contradiction must be completely inside one level of the stratification), and one

also has to check that the negative atoms (since Q is safe, their arguments are fully determined, once some candidates for the images of the positive atoms under the homomorphism are found) do not match with any of the atoms in a canonical model of D and Σ (which is done separately for each negative atom). $\square$

Proof of Theorem 33. The result is proved in the same way as Theorem 6, which follows from Theorem 5. The main difference is that the finite part of the guarded chase forest is now computed for each level of a stratification, and that we now also have to check that the negative atoms cannot be homomorphically mapped to a canonical model. We first compute a stratification of Σ, which is possible in constant time. We then compute sets similar to the S_i's, $i \in \{0, \ldots, k\}$, of Theorem 32. But to obtain all side atoms, by Theorem 5, with a slightly larger depth, namely, $l = (n+1) \cdot \delta$, where $n = |Q^+| + 1$, $\delta = (2w)^w \cdot 2^{(2w)^{w \cdot |\mathcal{R}|}}$, and w is the maximal arity of a predicate in $\mathcal{R}$. By the proof of Theorem 6, this and the evaluation of Q^+ and all $Q^+ \cup \{\mathbf{a}\}$, where $\mathbf{a} \in Q^-$, over S_k is possible in polynomial time. $\square$

Proof of Theorem 34. We use the same line of argumentation as in the proofs of Theorem 9 and Corollary 10, except that we now determine first a stratification μ of Σ and then iteratively (for each possible collection of database atoms with nulls as arguments) the guarded chase forest of bounded depth (which depends only on Q and $\mathcal{R}$) for every level of μ, and we also check that the negative atoms do not match with any of the atoms in the thus generated canonical model. $\square$

Proof of Lemma 35. Since $M \subseteq HB_\Sigma$ is an isomorphic image of both M_1^f and M_2^f, it follows that M_1^f and M_2^f are isomorphic. Similarly, also N_1^f and N_2^f are isomorphic, and the two subrelations of $\prec$ that are obtained from $\prec$ by restriction to $M_1^f \times N_1^f$ and $M_2^f \times N_2^f$ are isomorphic. $\square$

Proof of Proposition 36. ($\Rightarrow$) Let M be a perfect model of D and Σ. That is, (i) $M \subseteq HB_\Sigma$ is an isomorphic image of a model M^f of D and Σ^f and (ii) $M \ll N$ for all isomorphic images $N \subseteq HB_\Sigma$ of models of D and Σ^f such that N is not isomorphic to M. Towards a contradiction, suppose that M^f is not a minimal model of D and Σ^f. That is, there exists a minimal model N^f of D and Σ^f such that $N^f \subset M^f$. Thus, $M^f - N^f \neq \emptyset$. However, since $\prec$ is empty, $M \ll N$ does not hold, and since N is not isomorphic to M, this contradicts M being a perfect model of D and Σ. This shows that M^f is a minimal model of D and Σ^f.

($\Leftarrow$) Let M be an isomorphic image of the least model M^f of D and Σ^f, and let N be any isomorphic image of a model N^f of D and Σ^f such that N is not isomorphic to M. Since $M^f \subset N^f$, it follows that $M^f - N^f = \emptyset$. Hence, since M

is not isomorphic to N, it holds that $M \ll N$. This shows that M is a perfect model of D and Σ. $\square$

Proof of Proposition 37. Consider any $i \in \{0, 1, \ldots, k-1\}$.

($\Rightarrow$) Suppose that S' is a perfect model of $D^\star_{i+1}$ and $\Sigma^\star_{i+1}$. That is, (1) S' is an isomorphic image of a model M^f of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$ and (2) $S' \ll N$ for all isomorphic images $N \subseteq HB^\star_{i+1}$ of models of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$ such that N is not isomorphic to S'. Hence, (1) S is an isomorphic image of a model S^f of $D^\star_i$ and $(\Sigma^\star_i)^f$ and (2) $S \ll N$ for all isomorphic images $N \subseteq HB^\star_i$ of models of $D^\star_i$ and $(\Sigma^\star_i)^f$ such that N is not isomorphic to S. That is, (i) S is a perfect model of $D^\star_i$ and $\Sigma^\star_i$. Furthermore, as for (ii), since S' is an isomorphic image of a model M^f of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$, it follows that S' is an isomorphic image of a model M^f of $S^f \cup D_{i+1}$ and $(\Sigma^f_{i+1})^{S^f}$. We now show that M^f is also minimal. Towards a contradiction, suppose that M^f is not minimal. That is, there exists a minimal model N^f of $S^f \cup D_{i+1}$ and $(\Sigma^f_{i+1})^{S^f}$ such that $N^f \subset M^f$. It thus follows that $M^f - N^f \neq \emptyset$. Observe that N^f is also a model of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$. Since $M^f \cap (HB^\star_i)^f = N^f \cap (HB^\star_i)^f$, where $(HB^\star_i)^f$ is the natural extension of $HB^\star_i$ by function symbols, $S' \ll N$ does not hold, where N is an isomorphic image of N^f. But, since N is not isomorphic to S', this contradicts S' being a perfect model of $D^\star_{i+1}$ and $\Sigma^\star_{i+1}$. This shows that M^f is also minimal.

($\Leftarrow$) Suppose that (i) S is a perfect model of $D^\star_i$ and $\Sigma^\star_i$, and S is an isomorphic image of a model S^f of $D^\star_i$ and $(\Sigma^\star_i)^f$, and (ii) S' is an isomorphic image of a minimal model M^f of $S^f \cup D_{i+1}$ and $(\Sigma^f_{i+1})^{S^f}$. Thus, S' is also an isomorphic image of a model M^f of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$. We now show that S' is a perfect model of $D^\star_{i+1}$ and $\Sigma^\star_{i+1}$. That is, $S' \ll N$ for all isomorphic images N of models N^f of $D^\star_{i+1}$ and $(\Sigma^\star_{i+1})^f$ such that N is not isomorphic to S'. Recall that $S' \ll N$ iff for every $a \in M^f - N^f$, some $b \in N^f - M^f$ exists with $a \prec b$. Clearly, by (i), if $a \in (M^f - N^f) \cap (HB^\star_i)^f$, then ($\star$) some $b \in (N^f - M^f) \cap (HB^\star_i)^f$ exists with $a \prec b$. W.l.o.g., both M^f and N^f are minimal, and thus $M^f \cap (HB^\star_{i+1})^f$ and $N^f \cap (HB^\star_{i+1})^f$ are obtained from $S^f = M^f \cap (HB^\star_i)^f$ and $T^f = N^f \cap (HB^\star_i)^f$, respectively, by iteratively applying an immediate consequence operator via $(\Sigma^f_{i+1})^{S^f}$ and $(\Sigma^f_{i+1})^{T^f}$, respectively. Let $a_0, a_1, \ldots$ be the ordered sequence of all elements in $(M^f - N^f) \cap (HB^\star_{i+1})^f$ such that for every $i \in \{0, 1, \ldots\}$, it holds that a_i is derived before a_{i+1}. Then, $a_0 \in (M^f - N^f) \cap (HB^\star_{i+1})^f$ is justified either by some $a \in (M^f - N^f) \cap (HB^\star_i)^f$ with $a_0 \preccurlyeq a$ (as argued above, this implies ($\star$)) or by some $b \in (N^f - M^f) \cap (HB^\star_i)^f$ with $a_0 \prec b$. Similarly, every $a_i \in (M^f - N^f) \cap (HB^\star_{i+1})^f$ is justified either by some $a_j \in (M^f - N^f) \cap (HB^\star_{i+1})^f$ with $j \in \{0, 1, \ldots, i-1\}$ and $a_i \preccurlyeq a_j$, (by induction on $a_0, a_1, \ldots$, this implies ($\star$) with $a = a_j$),

by some $a \in (M^f - N^f) \cap (HB_i^\star)^f$ with $a_i \preccurlyeq a$ (as argued above, this implies $(\star)$), or by some $b \in (N^f - M^f) \cap (HB_i^\star)^f$ with $a_i \prec b$. In summary, this shows that for every $a \in M^f - N^f$, there exists some $b \in N^f - M^f$ such that $a \prec b$. $\square$

Proof of Theorem 39. Let $\mu\colon \mathcal{R} \to \{0, 1, \ldots, k\}$ be a stratification of Σ. Let $S_0 = chase(D, \Sigma_0)$ and $S_{i+1} = chase(S_i, \Sigma_{i+1}^{S_i})$ for $i \in \{0, 1, \ldots, k-1\}$. Recall that S_k is the canonical model of D and Σ. We now show by induction on $i \in \{0, 1, \ldots, k\}$ that $S_i - (D_{i+1} \cup \cdots \cup D_k)$ is a perfect model of $D_i^\star$ and $\Sigma_i^\star$. Hence, in particular, S_k is a perfect model of $D_k^\star = D$ and $\Sigma_k^\star = \Sigma$.

Basis: By Theorem 29, $S_0 - (D_1 \cup \cdots \cup D_k)$ is an isomorphic image of the least model of D_0 and Σ_0^f. By Proposition 36, it thus follows that $S_0 - (D_1 \cup \cdots \cup D_k)$ is a perfect model of $D_0 = D_0^\star$ and $\Sigma_0 = \Sigma_0^\star$.

Induction: By the induction hypothesis, $S_i' = S_i - (D_{i+1} \cup \cdots \cup D_k)$ is a perfect model of $D_i^\star$ and $\Sigma_i^\star$, which also implies that S_i' is an isomorphic image of a model S_i^f of $D_i^\star$ and $(\Sigma_i^\star)^f$. By Theorem 29, S_{i+1} is an isomorphic image of the least model of S_i and $(\Sigma_{i+1}^f)^{S_i}$. Thus, $S_{i+1}' = S_{i+1} - (D_{i+2} \cup \cdots \cup D_k)$ is an isomorphic image of the least model of $S_i' \cup D_{i+1}$ and $(\Sigma_{i+1}^f)^{S_i'}$. Hence, S_{i+1}' is also an isomorphic image of the least model of $S_i^f \cup D_{i+1}$ and $(\Sigma_{i+1}^f)^{S_i^f}$. By Proposition 37, S_{i+1}' is a perfect model of $D_{i+1}^\star$ and $\Sigma_{i+1}^\star$. $\square$

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *J. Comput. Syst. Sci.*, 43(1):62–124, 1991.

[3] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: Querying ontologies. In *Proc. AAAI-2005*, pp. 1670–1671, 2005.

[4] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *J. Phil. Log.*, 27(3):217–274, 1998.

[5] G. Antoniou. Non-monotonic rule systems on top of ontology layers. In *Proc. ISWC-2002*, pp. 394–398, 2002.

[6] K. R. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 89–148. Morgan Kaufmann, 1988.

[7] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. *DL-Lite* in the light of first-order logic. In *Proc. AAAI-2007*, pp. 361–366, 2007. Extended report: The *DL-Lite* family and relations. School of Computer Science and Information Systems, Birkbeck College, 2009.

[8] F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proc. IJCAI-2003*, pp. 319–324, 2003.

[9] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. IJCAI-2005*, pp. 364–369, 2005.

[10] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL – a polynomial-time reasoner for life science ontologies. In *Proc. IJCAR-2006*, pp. 287–291, 2006.

[11] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. Extending decidable cases for rules with existential variables. In *Proc. IJCAI-2009*, pp. 677–682, 2009.

[12] V. Barany, G. Gottlob, and M. Otto. Querying the guarded fragment. In *Proc. LICS-10*, pp. 1–10, 2010.

[13] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proc. ICALP-1981*, pp. 73–85, 1981.

[14] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[15] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Sci. Am.*, 284:34–43, 2001.

[16] D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema, 2004. W3C Recommendation.

[17] J. D. Bruijn, T. Eiter, A. Polleres, and H. Tompits. Embedding non-ground logic programs into autoepistemic logic for knowledge base combination. In *Proc. IJCAI-2007*, pp. 304–309, 2007.

[18] L. Cabibbo. The expressive power of stratified logic programs with value invention. *Inf. Comput.*, 147(1):22–56, 1998.

[19] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. ER-2001*, pp. 270–284, 2001.

[20] A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR-2008*, pp. 70–80, 2008. Revised version: http://benner.dbai.tuwien.ac.at/staff/gottlob/CGK.pdf.

[21] A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. In *Proc. PODS-2009*, pp. 77–86, 2009.

[22] A. Calì, G. Gottlob, and A. Pieris. Tractable query answering over conceptual schemata. In *Proc. ER-2009*, pp. 175–190, 2009.

[23] A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proc. VLDB-10*, 3(1):554–565, 2010.

[24] A. Calì, G. Gottlob, and A. Pieris. Query answering under expressive Entity-Relationship schemata. In *Proc. ER-10*, pp. 347–361, 2010.

[25] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in Datalog+/-. In *Proc. RR-10*, pp. 1–17, 2010.

[26] A. Calì and M. Kifer. Containment of conjunctive object meta-queries. In *Proc. VLDB-2006*, pp. 942–952, 2006.

[27] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. PODS-2003*, pp. 260–271, 2003.

[28] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. IJCAI-2003*, pp. 16–21, 2003.

[29] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic *DL-Lite*$_{\mathcal{A}}$. In *Proc. OWLED-2006*, 2006.

[30] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Can OWL model football leagues? In *Proc. OWLED-2007*, 2007.

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

[32] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. STOC-1977*, pp. 77–90, 1977.

[33] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. Comput.*, 14:671–677, 1985.

[34] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

[35] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[36] J. de Bruijn and S. Heymans. Logical foundations of (e)RDF(S): Complexity and reasoning. In *Proc. ISWC-2007*, pp. 86–99, 2007.

[37] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *Proc. PODS-2008*, pp. 149–158, 2008.

[38] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.

[39] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artif. Intell.*, 172(12/13):1495–1539, 2008.

[40] T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Trans. Comput. Log.*, 12(2), 2011.

[41] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proc. IJCAI-2005*, pp. 90–96, 2005.

[42] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for Semantic Web reasoning. In *Proc. ESWC-2006*, pp. 273–287, 2006.

[43] T. Eiter and M. Simkus. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.*, 11(2), 2010.

[44] R. Fagin. A normal form for relational databases that is based on domains and keys. *ACM Trans. Database Syst.*, 6(3):387–415, 1981.

[45] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[46] B. Faltings and S. Macho-Gonzalez. Open constraint programming. *Artif. Intell.*, 161(1/2):181–208, 2005.

[47] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *Proc. WWW-2003*, pp. 48–57, 2003.

[48] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. KR-2006*, pp. 57–67, 2006.

[49] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL rules: A proposal and prototype implementation. *J. Web Sem.*, 3(1):23–40, 2005.

[50] D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

[51] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[52] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. Combined FO rewritability for conjunctive query answering in *DL-Lite*. In *Proc. DL-2009*, 2009.

[53] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. KR-10*, 2010.

[54] A. Krisnadhi and C. Lutz. Data complexity in the $\mathcal{EL}$ family of DLs. In *Proc. DL-2007*, 2007.

[55] M. Krötzsch and S. Rudolph. Conjunctive queries for $\mathcal{EL}$ with composition of roles. In *Proc. DL-2007*, 2007.

[56] M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proc. ISWC-2008*, pp. 649–664, 2008.

[57] M. J. Lawlwy and C. Bousquet. Fast classification in Protégé: Snorocket as an OWL2 $\mathcal{EL}$ reasoner. In *Proc. DL-2010*, 2010.

[58] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS-2002*, pp. 233–246, 2002.

[59] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artif. Intell.*, 104(1/2):165–209, 1998.

[60] V. Lifschitz. Non-monotonic databases and epistemic queries. In *Proc. IJCAI-1991*, pp. 381–386, 1991.

[61] T. Lukasiewicz. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Trans. Knowl. Data Eng.*, 22(11):1577–1592, 2010.

[62] T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Approx. Reas.*, 45(2):288–307, 2007.

[63] T. Lukasiewicz. Fuzzy description logic programs under the answer set semantics for the Semantic Web. *Fundam. Inform.*, 82(3):289–310, 2008.

[64] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. IJCAI-2009*, pp. 2070–2075, 2009.

[65] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.

[66] D. Mailharrow. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12(4):383–397, 1998.

[67] B. Marnette. Generalized schema-mappings: from termination to tractability. In *Proc. PODS-2009*, pp. 13–22, 2009.

[68] B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *Proc. IJCAI-2007*, pp. 477–482, 2007.

[69] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[70] P. F. Patel-Schneider and I. Horrocks. A comparison of two modelling paradigms in the Semantic Web. *J. Web Sem.*, 5(4):240–250, 2007.

[71] H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Applied Logic*, 8(2):186–209, 2010.

[72] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[73] R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pp. 55–76. Plenum Press, 1978.

[74] R. Rosati. Towards expressive KR systems integrating Datalog and description logics: preliminary report. In *Proc. DL-1999*, 1999.

[75] R. Rosati. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.*, 3(1):61–73, 2005.

[76] R. Rosati. $\mathcal{DL}+\log$: Tight integration of description logics and disjunctive Datalog. In *Proc. KR-2006*, pp. 68–78, 2006.

[77] R. Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In *Proc. PODS-2006*, pp. 356–365, 2006.

[78] R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc DL-2007*, 2007.

[79] S. Rudolph, M. Krötzsch, and P. Hitzler. All elephants are bigger than all mice. In *Proc. DL-2008*, 2008.

[80] M. Sintek and S. Decker. TRIPLE – a query, inference, and transformation language for the Semantic Web. In *Proc. ISWC-2002*, pp. 364–378, 2002.

[81] M. Y. Vardi. On the complexity of bounded-variable queries. In *Proc. PODS-1995*, pp. 266–176, 1995.

[82] K. Wang, G. Antoniou, R. W. Topor, and A. Sattar. Merging and aligning ontologies in dl-programs. In *Proc. RuleML-2005*, pp. 160–171, 2005.

[83] F. Yang, X. Chen, and Z. Wang. P-dl-programs: Combining dl-programs with preferences for the Semantic Web. Unpublished manuscript, August 2006.