On Equality Constraints in Datalog+/- Knowledge Bases (Position Paper)

Andrea Calí[1], Marco Console[2] and Riccardo Frosini[3]

[1]University of Naples "Federico II", Italy
[2]Sapienza University of Rome, Italy
[3]Birkbeck University of London, UK

Abstract

Commonly adopted database constraints in knowledge bases are tuple-generating dependencies (TGDs) and equality-generating dependencies (EGDs), which form the core of the Datalog+/- family of languages, which are able to capture a variety of ontology formalisms. The presence of EGDs in Datalog+/- programs, and even in simpler relational schema languages based e.g. on inclusion dependencies, is known to easily lead to undecidability or intractability of query answering. The notion of separability was hence introduced to characterise sets of EGDs that have limited interaction with TGDs. We review two notions of separability found in the literature, as well as syntactic conditions that are sufficient to them. In particular, we define the notion of deep separability, which captures several analogous notions defined ad-hoc in the literature, and provide a sufficient condition for it. We then establish a relationship between decidability of query answering and decidability of determining separability. This works sheds light on the interaction between EGDs and TGDs in Datalog+/- knowledge bases and beyond, providing a basis for further investigations.

Keywords

Ontology Based Data Access, Semantic Technologies, Integrity Constraints

1. Introduction and Preliminaries

When a database D is equipped with an *ontology* Σ, that is, a knowledge base consisting of constraints that express relevant properties of the underlying domain, queries are not answered merely against the database instance D, but against the logical theory $D \cup \Sigma$. Several languages have been proposed for ontologies, with different computational properties.

The *DL-Lite* family [1] has the advantage of a low (AC_0, which is contained in LOGSPACE) data complexity of conjunctive query answering and of knowledge base satisfiability.

The $\text{ER}^{\pm}$ family of ER-like languages [2], in particular, comprises several tractable (w.r.t. conjunctive query answering) ontology languages, which properly generalize the main languages of the DL-Lite family.

Another relevant, more general class of ontology languages, capable of capturing most DL-lite language, is the *Datalog*$^{\pm}$ family, that is, a family of rule-based languages derived from Datalog (see, e.g., [3] and references therein) whose rules are (function-free) Horn rules, possibly with existentially quantified variables in the head, called *tuple-generating dependencies (TGDs)*,

SEBD 2025: 33rd Symposium on Advanced Database Systems, 16th–19th June, 2025, Ischia, Italy
andreacali@unina.it (A. Calí); console@diag.uniroma1.it (M. Console); frosini.riccardo@gmail.com (R. Frosini)

enriched with functionality constraints in the form of *equality-generating dependencies (EGDs)*, and *negative constraints*, a form of denial constraints.

In this paper we focus on the interaction between TGDs and EGDs. A TGD is an first-order implication that forces the existence of tuples under certain conditions, and it is of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \psi(\mathbf{X}, \mathbf{Z})$, where $\varphi(\mathbf{X}, \mathbf{Y})$ and $\psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over a relational schema. An EGD forces equality of values under certain conditions, and it is of the form $\forall \mathbf{X}\, \varphi(\mathbf{X}) \rightarrow X_i = X_j$, where $\varphi(\mathbf{X})$ is a conjunction of atoms over a relational schema, and $\{X_i, X_j\} \subseteq \mathbf{X}$. To answer a query Q over an instance D and a set Σ of TGDs and EGDs, we could in principle "expand" D according to Σ, inferring all the entailed additional knowledge, and then evaluate Q against the obtained instance. Said expansion is called *chase* in the literature, for which we refer the reader to [4]. In the chase, atoms are added according to TGDs; in doing so, unknown values (those corresponding to the existentially-quantified variables of TGDs) are represented by *labelled nulls*, which are a sort of placeholder. The set of labelled nulls is denoted by Γ_N, while the set of constants is denoted by Γ. EGDs force the equality between a labelled null and a constant, or between two labelled nulls[1]. The application of an EGD can trigger the application of TGDs. The chase as a procedure applies one TGD, then all EGDs until no further EGD application is possible, and then starts again with an EGD application, and so on, possibly to the infinite. The chase is a *universal model* in this case, which entails that the correct answer to a query Q on a database D under a set Σ of TGDs and EGDs can be obtained by evaluating Q on the chase of (only in principle, as the result of the chase, also called chase for convenience of notation, can be infinite).

In the rest of this section we show two examples to define the notion of *separability*, which shall be introduced formally in the next section.

Example 1. *Consider the following set Σ of TGDs and EGDs (we omit universal quantifiers to avoid clutter):*

$$\begin{array}{llll} \sigma_1: & r_1(X) \rightarrow \exists Y\, r_2(X,Y) & \sigma_4: & r_4(X,Y) \rightarrow r_5(X,Y) \\ \sigma_2: & r_2(X,Y) \rightarrow r_3(X,Y) & \sigma_5: & r_5(X,Y) \rightarrow r_2(X,Y) \\ \sigma_3: & r_3(X,Y) \rightarrow r_4(X,Y) & \eta: & r_3(X,Y), r_3(X,Z) \rightarrow Y = Z \end{array}$$

Notice that η is a key dependency*, imposing that atoms in r_3 have unique values on their first attribute. Now, let us take $D = \{r_1(a), r_3(a,b)\}$ and the (ground) Boolean conjunctive query Q defined as $q \leftarrow r_2(a,b)$ (q is a propositional predicate), which simply asks whether $r_2(a,b)$ holds. Let us answer Q by computing the chase of D according to Σ, denoted $chase(D,\Sigma)$. We first obtain $r_2(a,\zeta)$ from σ_1, where ζ is a labelled null. From σ_3 we get $r_4(a,b)$. We then add $r_3(a,\zeta)$ from σ_2; at this point we need to apply η and enforce $\zeta = b$, so that $r_3(a,\zeta)$ becomes $r_3(a,b)$. Due to η, therefore, the query has positive answer, written $D \cup \Sigma \models Q$. However, even in the absence of η, Q would be answered positively. In fact, if we proceed with the expansion we get $r_5(a,b)$ from σ_4 and finally $r_2(a,b)$ from σ_5. Therefore we would have had the same result without η. This can be shown to hold for every query Q; therefore,* provided that the theory $D \cup \Sigma$ is satisfiable *(that is, the expansion does not fail), we can answer every query, for every D, by considering the TGDs only. This property is called* separability*, and it defines a form of lack of interaction (hence the name) between EGDs and TGDs in a certain set.*

[1]Equating two distinct constants results in a logical inconsistency because the *unique name assumption* is adopted.

Example 2. *Suppose we have $D = \{r(a, b)\}$, a TGD $r(X, Y) \rightarrow \exists Z\, s(Y, Z)$, and an EGD $s(X_1, X_2), r(X_3, X_2) \rightarrow X_1 = X_2$. The chase (according to the dependencies above) then adds $s(b, \zeta)$ by virtue of the TGD ζ turns into b and the added atom becomes $s(b, b)$. Hence a query Q defined as $Q \rightarrow s(X, X)$ has positive answer, and would have negative answer without the EGD above. In this case separability does not hold.*

2. What is separability?

It is well know that the interaction of TGDs and EGDs leads to undecidability of query answering; this happens even in simple sub-classes of constraints, such as key and inclusion dependencies [5, 6]. It is therefore useful to identify classes of constraints which do not suffer from his "harmful" interaction. To this aim, a key condition is that of *separability* [2, 7], which we give below.

Definition 1 (Separability). *Consider a Σ_T of TGDs over a schema $\mathcal{R}$, and a set Σ_E of EGDs over $\mathcal{R}$. We say that the set $\Sigma = \Sigma_T \cup \Sigma_E$ is* separable *if, for every database D for $\mathcal{R}$, either $chase(D, \Sigma)$ fails, or, $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$, for every BCQ Q over $\mathcal{R}$.*

Notably, there is another definition of separability in the literature, which is adopted in [5, 6, 8]. Such a definition is similar to the above Definition 1, but with a difference which makes it stronger. For the sake of completeness, we give such a definition below.

Definition 2 (Old separability). *Consider a Σ_T of TGDs over a schema $\mathcal{R}$, and a set Σ_E of EGDs over $\mathcal{R}$. We say that the set $\Sigma = \Sigma_T \cup \Sigma_E$ is* separable *if, for every database D for $\mathcal{R}$, (i) if the chase fails, then $D \not\models \Sigma_E$ (D does not satisfy Σ), and (ii) if the chase does not fail, we have $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$, for every BCQ Q over $\mathcal{R}$.*

The old separability is a special case of the new separability as it enforces condition *(i)* (Definition 2), which we reformulate below, calling it *EGD-stability*.

Definition 3 (EGD-stability). *Consider a set Σ_T of TGDs over a schema $\mathcal{R}$, and a set Σ_E of EGDs over $\mathcal{R}$. We say that the set $\Sigma = \Sigma_T \cup \Sigma_E$ is* EGD-stable *if, for every instance D for $\mathcal{R}$, $D \models \Sigma_E$ implies that $chase(D, \Sigma)$ does not fail.*

EGD stability guarantees that the satisfiability of $D \cup \Sigma$ (i.e., the existence of $chase(D, \Sigma)$) can be determined by merely checking whether $D \models \Sigma_E$. The satisfiability check is a fundamental step in query answering (see Section 2.2), and EGD stability guarantees it can be easily done. However, with a more sophisticated version of the satisfiability check (see Section 3), the old separability can be relaxed to the new separability, giving raise to the discovery of new (syntactic) classes that enjoy (new) separability. Section 2.1 below provides an overview of the most relevant syntactic classes in the literature.

2.1. Related Work

The notion of (old) separability was first introduced in [5, 6], in the context of inclusion dependencies (IDs) and key dependencies (KDs) (see e.g. [9]). The general idea is to define a

sufficient syntactic condition for separability, which can be efficiently checked. This was done while extending an early class of IDs and KDs (see e.g. [9]), called *key-based*. Key-based IDs and KDs were proposed in the milestone work by Johnson and Klug [10], and they are in fact separable, though not defined explicitly as such. Key-based IDs and KDs are defined as follows[2]: *(i)* for each relational predicate r, there is *only one KD* defined on it; *(ii)* for each ID $r[\mathbf{X}] \subseteq s[\mathbf{Y}]$, where $\mathbf{X}$ and $\mathbf{Y}$ are set of attributes of r and s respectively (see [9] for the notation), *(ii.a)* $\mathbf{X}$ is *disjoint* from any key set of attributes for r, and *(ii.b)* $\mathbf{Y}$ is *properly contained* in the set of key attributes for s.

The more general class of *non-key-conflicting (NKC) IDs*, with respect to a set of KDs, is defined as follows: *(i)* for each relational predicate r, there is *only one KD* defined on it; *(ii)* for every ID $r[\mathbf{X}] \subseteq s[\mathbf{Y}]$, $\mathbf{Y}$ is *not* a *proper* superset of the set of key attributes for s (if such set exists). The class of KDs and NKC IDs is separable, and it properly captures the well known class of *foreign-key dependencies*.

The class of KDs and non-key conflicting IDs was generalized in [8] to general TGDs, which are assumed to have a single atom in the head, without loss of generality. The idea is analogous, and the condition is as follows: *(i)* for each relational predicate r, there is *only one KD* defined on it; *(ii)* each existentially quantified variable in the head of a TGD must occur only once; *(iii)* for each TGD $\sigma = \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, r(\mathbf{X}, \mathbf{Z})$, the set of $\mathbf{X}$-attributes of $r(\mathbf{X}, \mathbf{Z})$ is not a proper superset of the set of key attributes of r.

In [11], the class of non-key-conflicting TGDs was straightforwardly extended to treat functional dependencies (FDs) rather than keys.

The literature so far discussed deals with sufficient *syntactic* conditions that guarantee, in fact, EGD-stability. However, there are classes of TGDs and EGDs such that EGDs are triggered in the chase, but which enjoy separability.

Example 3. *Consider the set of TGDs and EGDs in Example 1. It is separable, but it is not EGD-stable. In fact, if we consider the instance* $D' = \{r_3(a, b), r_2(a, c)\}$*, we have that* $D' \models \Sigma_E$ *but* $D \cup \Sigma_E$ *is unsatisfiable because the chase fails due to a hard violation.*

This leads in [2] to the definition of separability as in Definition 1 in this paper. This work deals with IDs and KDs which express an expressive variant of the Entity-Relationship model [12]. By means of graph-related properties, *necessary and sufficient* syntactic conditions for separability are provided, thus defining useful tractable classes of constraints.

The case of *linear TGDs* (TGDs with a single body-atom) and KDs was later considered in [13, 7]. In these works, sufficient syntactic conditions are proposed that guarantee separability, without imposing non-egd-triggerability. The conditions are quite involved and they make use of backward-resolution. Interestingly, the complexity of checking the syntactic condition, called *non-conflict* condition, is the same as that of query answering, that is PSPACE-complete.

At this point, we considered separability but not the problem of satisfiability. While separability allows us to ignore EGDs in the case of satisfiable theories, the problem of deciding the satisfiability remains, as well as that of determining its complexity. This will be the subject of next section.

[2]The definition in the original paper is different but equivalent; we choose a definition which is more clear in the context of this paper.

2.2. Query Answering under Separable Constraints

In the case of a set $\Sigma = \Sigma_T \cup \Sigma_E$ of separable TGDs and EGDs, such that BCQ answering under Σ_T is decidable, given an instance D and a BCQ Q, to decide whether $D \cup \Sigma \models Q$, the following steps are needed:

1. Check whether the chase fails, that is, whether $D \cup \Sigma$ is satisfiable; if $D \cup \Sigma$ is unsatisfiable, then trivially $D \cup \Sigma \models Q$ (*"Ex falso quodlibet"*).
2. If $D \cup \Sigma$ is satisfiable, then by Definition 1 we know $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$, therefore we check whether $chase(D, \Sigma_T) \models Q$.

Apart from the complexity of the satisfiability check, we have that the complexity of query answering is the same as that of answering under TGDs only, which is a highly desirable property. In the case of EGD-stable constraints, the satisfiability check amounts simply to checking whether $D \models \Sigma_E$, which can be done in NP, and in PTIME (or better, in the even lower complexity class AC_0) if we consider Σ_E fixed. However, in the cases of separable but not EGD-stable constraints, the problem is to be addressed differently; this will be the subject of Section 3.

3. Separability and Satisfiability

In this section we address the problem of deciding whether, given an instance D and a set Σ of *separable* TGDs and EGDs, $D \cup \Sigma$ is satisfiable. As seen in Section 2, this preliminary check is needed, in the case of separability, before one proceeds to answer a BCQ Q by taking into account the TGDs only.

The satisfiability check is done as in [8, 2], by encoding *hard violations* of EGDs as a set $\mathcal{Q}_V$ of Boolean conjunctive queries. Given a separable set $\Sigma = \Sigma_T \cup \Sigma_E$, and an instance D, we have that satisfiability holds if and only if, for each $Q_V \in \mathcal{Q}_V$ we have $D \cup \Sigma_T \not\models Q_V$ or, equivalently, if and only if all queries in $\mathcal{Q}_V$ have negative answer when evaluated against D and Σ_T (TGDs only). The encoding is done as follows. First, we need to add auxiliary facts to D. For each pair of *distinct* constants c_1, c_2 appearing in D as arguments, we add the facts $neq(c_1, c_2)$ and $neq(c_2, c_1)$ to D, where $neq/2$ is an auxiliary predicate expressing that two constants are distinct. Then, for each EGD η of the form $\phi(\mathbf{X}) \rightarrow X_i = X_j$, with X_i and X_j in $\mathbf{X}$, we construct the BCQ Q_η as follows (quantifiers omitted for brevity): $Q_\eta : q \leftarrow \phi(\mathbf{X}), neq(X_i, X_j)$, where q is a propositional predicate. It is not difficult to prove that if $D \cup \Sigma \models Q_\eta$ then $chase(D, \Sigma)$ fails, and therefore $D \cup \Sigma$ is unsatisfiable. However, it still remains to determine whether $D \cup \Sigma \models Q_\eta$. Notice that in the case of non-egd-triggerable constraints we do not have this problem, as we merely need to check whether $D \models \Sigma_E$. In principle, separability (see Definition 1 does not tell us anything about the cases when the chase fails. However, we are still in luck because we can evaluate the (Boolean) conjunctive queries in Q_V against $D \cup \Sigma_T$ rather than $D \cup \Sigma$. This is proved, for instance, in [2, 13], but *ad hoc*, by using the properties of the particular class of constraints involved. Here we show a general condition that allows us to perform the satisfiability check as above. We first need to introduce a preliminary condition, called *deep separability*, which is apparently more restrictive than separability (we shall then prove that it is implied by separability). We start by denoting by $chase^k(D, \Sigma)$ the result of the first k chase steps under Σ applied to an instance D, where a step is either an EGD or a TGD application.

Definition 4. *Let Σ be a set of TGDs and EGDs, with $\Sigma = \Sigma_T \cup \Sigma_E$, where the dependencies in Σ_T are TGDs and those in Σ_E are EGDs. We say that Σ is* deeply separable *if, for each integer $k \geqslant 0$, for each instance D with with values in $\Gamma \cup \Gamma_N$, and for each Boolean conjunctive query Q, the following holds: if $chase^k(D, \Sigma)$ exists, then $chase^k(D, \Sigma) \models Q$ implies $chase(D, \Sigma_T) \models Q$.*

The notion of *deep separability*, intuitively, guarantees that, at any step before a possible failure, the chase does not entail any atoms that are not entailed by the chase computed according to Σ_T only. We now come to the main result about deeply separable TGDs and EGDs. The result states that in the case of deep separability the satisfiability check done by answering suitable queries as above is correct and complete.

Theorem 1. *Let Σ be a set of deeply separable TGDs and EGDs, with $\Sigma = \Sigma_T \cup \Sigma_E$, where the dependencies in Σ_T are TGDs and those in Σ_E are EGDs. Let $\mathcal{Q}_V = \{Q_1, \ldots, Q_m\}$ be the set of BCQs encoding violations of the EGDs in Σ_E as from the above construction. Then we have that $chase(D, \Sigma_T) \models Q_i$ for some $i \in \{1, \ldots, m\}$ if and only if $chase(D, \Sigma)$ fails (or equivalently $D \cup \Sigma$ is unsatisfiable).*

Proof (sketch). We prove the two directions of the implication separately.

"Only if". By contradiction, assume $chase(D, \Sigma_T) \not\models Q_i$ for some $i \in \{1, \ldots, m\}$ but $chase(D, \Sigma)$ exists. It is not difficult to show that if $chase(D, \Sigma_T) \not\models Q_i$ then also $chase(D, \Sigma) \not\models Q_i$; this holds because $chase(D, \Sigma_T)$ is a universal model for $D \cup \Sigma_T$ and $chase(D, \Sigma)$ is a model (not necessarily universal) for $D \cup \Sigma_T$; therefore there exists a homomorphism from the former to the latter. If $chase(D, \Sigma) \not\models Q_i$ then the chase must necessarily fail. Contradiction.

"If". Assume $chase(D, \Sigma)$ fails at step k by violation of the EGD $\eta \in \Sigma_E$. Then, $chase^{k-1}(D, \Sigma)$ exists; moreover, it is easily seen that $chase^{k-1}(D, \Sigma) \not\models Q_\eta$, where $Q_\eta \in \mathcal{Q}_V$ encodes the violation of η. By the hypothesis of deep separability we have $chase(D, \Sigma_T) \not\models Q_\eta$, hence the claim. □

Notice that for the "If" direction we do not need deep separability nor separability; this direction of the implication holds for general TGDs and EGDs. Finally we show that, despite the appearances, deep separability is not a stricted condition than separability. In fact, separability implies deep separability.

Theorem 2. *Let Σ be a set of TGDs and EGDs. If Σ is separable, then it is deeply separable.*

Proof (sketch). We distinguish two cases.

Case 1: $chase(D, \Sigma)$ exists. In this case the claim obviously holds.

Case 2: $chase(D, \Sigma)$ fails. Assume $chase(D, \Sigma)$ fails at step k; therefore $chase^{k-1}(D, \Sigma)$ exists. Take D_F obtained from D by replacing each constant in Γ with a fresh null from Γ_N. Obviously $chase(D_F, \Sigma)$ exists and so does $chase^{k-1}(D_F, \Sigma)$. Take any BCQ Q such that $chase^{k-1}(D_F, \Sigma) \not\models Q$. It is straightforwardly seen that $chase(D_F, \Sigma) \not\models Q$ and, due to separability, $chase(D_F, \Sigma_T) \not\models Q$. Since $chase(D_F, \Sigma_T) \not\models Q$ is obtained from $chase(D_F, \Sigma_T)$ by the above renaming of constants into nulls, we have $chase(D, \Sigma_T) \not\models Q$.

Since this holds for every step $\ell \leqslant k-1$, the claim is proved. □

The following result immediately follow from the above.

Corollary 1. *For every separable set Σ of TGDs and EGDs, for every instance D and for every BCQ Q:*

- *checking unsatisfiability of $D \cup \Sigma$ has the same complexity as query answering under Σ_T alone;*
- *checking whether $D \cup \Sigma \models Q$ has the same complexity as query answering under Σ_T alone.*

Notice that we mention *unsatisfiability* rather than satisfiability because Theorem 1 shows a reduction from failure (unsatisfiability) to BCQ answering under TGDs alone.

3.1. A Syntactic Condition for Separability

We consider *positions* in the schema as arguments of predicates; if we have a predicate $r/3$ of arity 3, we denote its arguments as positions $r[1]$, $r[2]$ and $r[3]$. Given a set Σ_T of TGDs, we say that a position is a *bud position* if, whenever it appears in a head-atom of any TGD of Σ_T, it contains an existentially-quantified variable. Given an EGD body $\varphi(\mathbf{X})$, we denote by $\varphi(\mathbf{X})|_{X_1/X_2}$, with $\{X_1, X_2\} \subseteq \mathbf{X}$, the conjunction of atoms obtained from $\varphi(\mathbf{X})$ by replacing X_1 with X_2. Given an EGD $\varphi(\mathbf{X}) \rightarrow X_1 = X_2$, we call X_1, X_2 *equality variables (EVs)*.

Definition 5. *Given a set $\Sigma = \Sigma_T \cup \Sigma_E$ as before, we say that Σ is* transparent *if: (i) for every EGD $\varphi(\mathbf{X}) \rightarrow X_1 = X_2$ in Σ_E, the equality variables appear in the body* only *in bud positions, and (ii) every atom $\underline{a}$ in $\varphi(\mathbf{X})|_{X_1/X_2}$ or $\varphi(\mathbf{X})|_{X_2/X_1}$ is such that $\varphi(\mathbf{X}) \cup \Sigma_T \models \underline{a}$.*

Intuitively, transparency ensures that *(i)* whenever an EGD is applied in the chase, the equation of two symbols (two labelled nulls, or one labelled null and a constant) does not propagate backwards in the portion of the chase constructed at that point; in fact, any labelled null corresponding to an EV in the application appears in the chase for the first time; *(ii)* all atoms appearing as a consequence of the EGD application would appear anyway in $chase(D, \Sigma_T)$. The following theorem can then be proved.

Theorem 3. *If a set $\Sigma = \Sigma_T \cup \Sigma_E$ of TGDs and EGDs is transparent, then Σ is separable.*

3.2. Separability and Decidability

To establish a relationship between the problem of determining the separability of a set of TGDs plus EGDs and decidability of conjunctive query answering, we show the following result.

Theorem 4. *Consider a set $\Sigma = \Sigma_T \cup \Sigma_E$ of TGDs and EGDs as before. Determining whether Σ is separable is undecidable.*

The above theorem can be proved by reducing the problem of conjunctive query answering to the one of checking separability.

4. Discussion

In this paper we have given an overview of the problem of separability between TGDs and EGDs in the context of ontological query answering, where queries are (Boolean) conjunctive queries. We have reviewed the two main notions of separability found in the literature, the "old" one and the "new" one, and we have clarified the difference between them. We have then addressed the issue of checking satisfiability of a set of TGDs and EGDs together with an instance, and we have shown that this can be done by merely answering suitable queries in the case of deeply separable classes of constraints. We have shown the desirable property that all separable sets of constraints are also deeply separable. We have therefore clarified and proved formally a satisfiability check which was already employed in the literature, but proved on a case-by-case basis [7, 8, 2], depending on the class of constraints. We believe that our generalisation provides a useful tool for future studies, and a better insight into the satisfiability problem.

Further results. More results on this paper's topics (for which we also refer the reader to an older paper [14]), which we did not include here due to space limitations, will be published elsewhere. In [7] a sufficient condition for separability (in the "new" meaning) is provided for the case of EGDs and linear TGDs (which, we remind the reader, are TGDs with exactly one head-atom and one body-atom), with ontologies that encode so-called *Extended Entity-Relationship* schemata. Such a condition can be extended (with some changes) to the class of *sticky sets of TGDs* [11], a relevant class that allows for a form of joins in the body.

Open problems. We currently have two main open problems at hand, which seem non-trivial. First, we would like to study the decidability of determining whether a given set of TGDs and EGDs is separable. This has been proved to be undecidable for the case of arbitrary TGDs and EGDs [15]; however, the proof relies on the fact that query answering under arbitrary TGDs (without EGDs) is undecidable [16]. It is not clear whether undecidability holds also in the cases where query answering is undecidable under TGDs and EGDs together, but decidable under TGDs alone; relevant cases are, for instance, IDs and KDs, sticky sets of TGDs and EGDs, or guarded TGDs and KDs [17]. We conjecture that determining separability is undecidable for these classes. The second problem is more technical, and is determining the complexity of checking the aforementioned, syntactic condition for the separability of sticky sets of TGDs and EGDs. We have an obvious EXPTIME lower bound, but the upper bound is currently unknown.

Ackowledgments. Andrea Calí acknowledges financial support from: project SERICS (PE00000014); "cascade" project SMIMI (CUP F53C24000190005), under the NRRP MUR program funded by the EU-NGEU; PNRR ECS00000037 "cascade" project MUSA; project INFANT (CUP E23C24000390006); project MELODY (PRIN-PNRR, CUP E53D23017550001). The authors would like to thank Andreas Pieris for his insightful comments on this research.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations, J. Artif. Intell. Res. 36 (2009) 1–69.

[2] A. Calì, G. Gottlob, A. Pieris, Ontological query answering under expressive Entity-Relationship schemata, Inf. Syst. 37 (2012) 320–335.

[3] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, A. Pieris, Datalog+/-: A family of logical knowledge representation and query languages for new applications, in: Proc. of LICS, 2010, pp. 228–242.

[4] A. Calì, M. Console, R. Frosini, Deep separability of ontological constraints, 2013. URL: https://arxiv.org/abs/1312.5914.

[5] A. Calì, Query Answering and Optimisation in Data Integration Systems, Ph.D. thesis, Università di Roma "La Sapienza", 2003.

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

[7] A. Calì, G. Gottlob, A. Pieris, Querying conceptual schemata with expressive equality constraints, in: Proc. of ER, 2011, pp. 161–174.

[8] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, J. Web Sem. (2012). To appear.

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

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

[11] A. Calì, G. Gottlob, A. Pieris, Advanced processing for ontological queries, PVLDB 3 (2010) 554–565.

[12] P. P. Chen, The entity-relationship model: towards a unified view of data, ACM TODS 1 (1995) 124–131.

[13] A. Calì, A. Pieris, On equality-generating dependencies in ontology querying – Preliminary report, in: Proc. of AMW, 2011.

[14] A. Calì, M. Console, R. Frosini, A. Pieris, On Equality Constraints in Ontological Query Answering, Technical Report, University of London, Birkbeck College, 2012. Available from the authors.

[15] A. Pieris, 2012. Personal communication.

[16] C. Beeri, M. Y. Vardi, The implication problem for data dependencies, in: Proc. of ICALP, 1981, pp. 73–85.

[17] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, in: Proc. of PODS, 2009, pp. 77–86.