Combining Entity Resolution and Query Answering in Ontologies: A Formal Conceptual Framework

(Discussion Paper)

Ronald Fagin[1], Phokion G. Kolaitis[1,2], Domenico Lembo[3], Lucian Popa[1] and Federico Scafoglieri[3,*]

[1] *IBM Research, Almaden, USA*
[2] *UC Santa Cruz, USA*
[3] *Sapienza University of Rome, Italy*

Abstract

This paper reports on a recent formal framework for integrating entity resolution and query answering within ontologies governed by rules expressed in the form of tuple-generating dependencies and equality-generating dependencies. Our framework outlines the semantics of the ontology by defining special instances involving equivalence classes of entities and sets of values. The former serve to gather entities referring to the same real-world object, and the latter to catalog alternative attribute values. This approach not only addresses entity resolution but also overcomes potential inconsistencies within the data. Additionally, we introduce a tailored chase procedure for this framework, ensuring non-failure and ultimately yielding a universal solution. This universal solution, in turn, can be used to obtain certain answers to conjunctive queries.

Keywords

Formal Conceptual Framework, Entity Resolution, Query Answering, Knowledge Bases, Ontology, Chase, Universal Model, tuple-generating dependencies, equality-generating dependencies, entity-resolution rules

1. Introduction

Entity resolution is the problem of determining whether different data records refer to the same real-world object, such as the same individual or the same organization, etc. [1, 2].

In this discussion paper we summarize a framework recently appeared in [3] for entity resolution in combination with query answering in the context of ontologies consisting of ground atoms and rules specified as tuple-generating dependencies (tgds) and equality-generating dependencies (egds). These rules have been widely investigated in databases and knowledge representation, e.g. in [4, 5, 6, 7]; they can express axioms that are used in Description Logics (DLs) [8], as well as in knowledge bases that are similar to Datalog +/- programs [9]. Additionally, egds are employed to express typical entity resolution rules that one may write in practice, as in [10]. These rules specify the conditions under which to entities have to be made equal, to say, for instance, that two persons are indeed the same person if they have similar names and the same date of birth. In

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy

[*] Corresponding author.

fagin@us.ibm.com (R. Fagin); kolaitis@ucsc.edu (P. G. Kolaitis); i.lembo@diag.uniroma1.it (D. Lembo); lpopa@us.ibm.com (L. Popa); scafoglieri@diag.uniroma1.it (F. Scafoglieri)

the following, we call such rules *entity-egds*, whereas we name *value-egds* the egds imposing equality on values, used, e.g., to express that a person has only one name.

In the proposed framework, the first challenge we face is to come up with a consistent way to complete or modify the original ontology, while respecting all its rules. To this aim we propose a new notion of valid models, called *ontology solutions*. They must satisfy all entity resolution rules along with all other ontology rules. Furthermore, the solutions must include all original data (i.e., no information is ever dropped or altered). Our approach is guided by the intuitive principle that for each real-world object there must be a single node in the solution that represents all "equivalent" entities denoting the object. To achieve this, we use equivalence classes of entities that become first-class citizens in our framework. In addition, we relax the standard way in which value-egds are satisfied by allowing solutions to use sets of values. Intuitively, we interpret egds as matching dependencies [11, 1, 12]. That is, when the conditions expressed by (the body of) an entity-egd or a value-egd hold in the data, and thus two entities or two values must be made equal, we combine them through a merging function that amounts to taking the union of all the alternatives. In this way, we group all entities that denote the same real-world object into a unique set, which turns out to be an equivalence class. Similarly, when two values exist where only one value is instead allowed according to the TBox, we explicitly form their union. Note that we consider the union of entities a "global" feature for a solution, that is, an entity e may belong to exactly one equivalence class; in contrast, the union of values is "local" to the context in which a value occurs, that is, a particular value may belong to more than one set of values.

We also remark that the semantics we adopt for value-egds allows us to always have a solution (that is, a model) for an ontology, even when there are no models according to the standard first-order semantics. Indeed, if a value-egd enforces the equality of different values, we collect together such values, whereas first-order logic would conclude that the ontology is inconsistent.

Besides formalizing the aforementioned ideas, our contribution is on how to combine entity resolution with query answering. In this respect, (i) we define *universal solutions* and show that, as for standard tgd and egd semantics, universal solutions can be used to obtain the certain answers of Conjunctive Queries (CQs); (ii) we propose a variant of the classical *chase* procedure [4, 5] tailored to our approach; (iii) we show that, when the chase procedure terminates, it returns a universal solution (and thus we have an algorithm for computing the certain answers to CQs).

In the following, we discuss our findings mainly by examples and refer the reader to [3] for a complete formal treatment.

2. Framework

We refer the reader to [13] for an introduction to equivalence classes and relations. In the following, let $\sim$ be an equivalence relation, we write $[x]$ to denote the equivalence class containing x, instead of $[x]_\sim$. Moreover, we may write, for example $[a, b, c]_\sim$ (or simply $[a, b, c]$) to denote the equivalence class consisting of the elements a, b, and c.

Syntax. An *atom* is an expression of the form $P(t_1, \ldots, t_n)$, where P is a symbol to denote an n-ary ontology or built-in predicate, i.e. a pre-interpreted predicate like the Jaccard similarity ($JaccSim$). Each t_i is either a variable or a constant, which can be in turn a value or an entity, the latter used to denote a real-world object. We impose that when P is a built-in predicate, constants occurring in $P(t_1, \ldots, t_n)$ are only values. A *ground* atom is an atom with no variables.

A *conjunction* $\phi(\boldsymbol{x})$ *of atoms* is an expression $P_1(\boldsymbol{t}_1) \wedge \ldots \wedge P_m(\boldsymbol{t}_m)$, where each $P_j(\boldsymbol{t}_j)$ is an atom such that each variable in the tuple $\boldsymbol{t}_j$ of terms is among those in the tuple $\boldsymbol{x}$ of variables.

An *ontology* $\mathcal{O}$ is a pair $(\mathcal{T}, \mathcal{A})$, consisting of a TBox $\mathcal{T}$ and a ABox $\mathcal{A}$. The TBox is a finite set of *tuple-generating dependencies* (tgds) and *equality-generating dependencies* (egds).

A *tgd* is a formula of the form:

$$\forall \boldsymbol{x}\, (\phi(\boldsymbol{x}) \to \exists \boldsymbol{y}\, \psi(\boldsymbol{x}, \boldsymbol{y})),$$

where $\phi(\boldsymbol{x})$ and $\psi(\boldsymbol{x}, \boldsymbol{y})$ are conjunctions of atoms, such that $\boldsymbol{x}$ and $\boldsymbol{y}$ have no variables in common and $\psi(\boldsymbol{x}, \boldsymbol{y})$ is built-in free and, for simplicity, constant-free. As in [5], we assume that all variables in $\boldsymbol{x}$ appear in $\phi(\boldsymbol{x})$, but not necessarily in $\psi(\boldsymbol{x}, \boldsymbol{y})$. We call $\phi(\boldsymbol{x})$ the body of the tgd, and $\psi(\boldsymbol{x}, \boldsymbol{y})$ the head of it.

An *egd* is a formula of the form:

$$\forall \boldsymbol{x}\, (\phi(\boldsymbol{x}) \to y = z),$$

where $\phi(\boldsymbol{x})$ is a conjunction of atoms and y and z are distinct variables occurring in $\phi(\boldsymbol{x})$, such that either both y and z are entity-variables (in which case we have an *entity-egd*) or both y and z are value-variables (in which case we have a *value-egd*). We call $\phi(\boldsymbol{x})$ the body of the egd.

In the examples, for simplicity, we will omit the universal quantification in tgds and egds.

Example 1. Let $\mathcal{T}$ be the TBox consisting of the rules:

$$\begin{array}{l}
(r_1)\ \mathit{Name}(p_1, n_1) \wedge \mathit{Name}(p_2, n_2) \wedge \mathit{JaccSim}(n_1, n_2, 0.7) \to p_1 = p_2 \\
(r_2)\ \mathit{Name}(p, n_1) \wedge \mathit{Name}(p, n_2) \to n_1 = n_2 \\
(r_3)\ \mathit{HPhone}(p, f_1) \wedge \mathit{HPhone}(p, f_2) \to f_1 = f_2 \\
(r_4)\ \mathit{HPhone}(p_1, f) \wedge \mathit{HPhone}(p_2, f) \to \mathit{SameHouse}(p_1, p_2, f)
\end{array}$$

Here p_1, p_2, p are *entity-variables*, i.e., occur in predicate arguments ranging over entities, whereas n_1, n_2, f_1, f_2, f are *value-variables*, i.e., occur in predicate arguments ranging over values. The predicates have the meaning suggested by their names. In particular, the predicate Name maintains the individual's name, HPhone stores their home phone contact, whereas $\mathit{SameHouse}$ associates individuals living in the same house with their phone number. Each of the four rules makes an assertion about the predicates. In particular, the first rule (r_1) states that if the Jaccard similarity of two names is higher then 0.7, then these names are associated to the same individual. The TBox also says that everyone can have only one name (r_2) and only one landline phone number (r_3), and that two individuals with the same landline phone number live in the same house (r_4). □

The ABox $\mathcal{A}$ of an ontology $\mathcal{O}$ is a finite set of ground atoms of the form $P(c_1, \ldots, c_n)$ where each c_i is an entity or a value.

Example 2. Let $\mathcal{A}$ be the ABox consisting of the atoms

$$\begin{array}{ll}
(g_1)\ \mathit{Name}(\mathsf{Doe}_1, \texttt{John Doe}), & (g_2)\ \mathit{HPhone}(\mathsf{Doe}_1, \texttt{358}) \\
(g_3)\ \mathit{Name}(\mathsf{Doe}_2, \texttt{Johnny Doe}), & (g_4)\ \mathit{HPhone}(\mathsf{Doe}_2, \texttt{635}) \\
(g_5)\ \mathit{Name}(\mathsf{Doe}_3, \texttt{Mary Doe}), & (g_6)\ \mathit{HPhone}(\mathsf{Doe}_3, \texttt{358})
\end{array}$$

In words, the ABox $\mathcal{A}$ specifies that Doe_1 has name `John Doe` and home phone number `358`, Doe_2 has name `Johnny Doe` and home phone number `635`, Doe_3 has name `Mary Doe` and phone number `358`. □

Semantics. We assume to have countably infinite many entity-nulls and value-nulls (note that we consider generic tgds, i.e. possibly having existential variables in the rule head, even though, for space limits we do not provide examples of this kind). The semantics of the ontology is given using special ABoxes, called ontology *instances*, whose ground atoms have components that are either equivalence classes of entities and entity-nulls, which we call *E-sets*, or non-empty sets of values and value-nulls, which we call *V-sets*.

***Ontology Instance* (Definition 1 of [3]).** An *instance $\mathcal{I}$ for an ontology $\mathcal{O}$* w.r.t. *an equivalence relation* $\sim$ is a set of *facts* $P(T_1, \ldots, T_n)$ such that P is a n-ary predicate in $\mathcal{O}$, and each T_i is either an E-set *w.r.t.* $\sim$ or a V-set.

Example 3. Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology such that $\mathcal{T}$ and $\mathcal{A}$ are as in Example 1 and in Example 2, respectively. Consider the following set $\mathcal{I}$ of facts.

(d_1) $\mathit{Name}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{John Doe}, \texttt{Johnny Doe}\})$ (d_2) $\mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}, \texttt{635}\})$
(d_3) $\mathit{Name}([\mathsf{Doe}_3], \{\texttt{Mary Doe}\})$ (d_4) $\mathit{HPhone}([\mathsf{Doe}_3], \{\texttt{358}\})$
(d_5) $\mathit{SameHouse}([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3])$ (d_6) $\mathit{SameHouse}([\mathsf{Doe}_3], [\mathsf{Doe}_1, \mathsf{Doe}_2])$

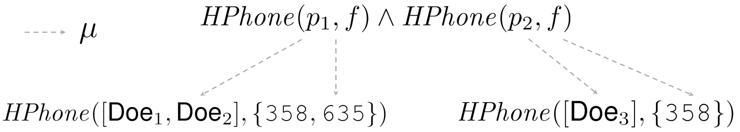
$\mathcal{I}$ is an instance for $\mathcal{O}$ w.r.t. the equivalence relation $\sim$ over the set $\{\mathsf{Doe}_1, \mathsf{Doe}_2, \mathsf{Doe}_3\}$ such that $\mathsf{Doe}_1 \sim \mathsf{Doe}_2$, i.e. Doe_1 and Doe_2 are in the same equivalence class and Doe_3 is the only element of another obviously disjoint equivalence class. □

To define the notions of satisfaction of *tgds* and *egds* by an instance, we first introduce the notion of an *assignment* from a conjunction $\phi(\boldsymbol{x})$ of atoms to an instance $\mathcal{I}$ for an ontology $\mathcal{O}$.

***Assignment* (Definition 3 of [3]).** An *assignment from a conjunction $\phi(\boldsymbol{x})$ of atoms to an instance $\mathcal{I}$* is a mapping μ from the variables and values in $\phi(\boldsymbol{x})$ to E-sets and V-sets of $\mathcal{I}$ such that

- each entity-variable is mapped to an E-set;
- different occurrences of a value-variable are mapped to V-sets with *non-empty intersection*;
- each value v is mapped to a V-set V, such that $v \in V$;
- for each atom $P(x, e, v)$ of $\phi(\boldsymbol{x})$, where x is a variable, e is an entity and v is a value, $\mathcal{I}$ contains a fact of the form $P(\mu(x), [e], \mu(v))$ (the definition generalizes in the obvious way for atoms of different form).

Example 4. Let $\mathcal{I}$ be the instance in Example 3. A possible assignment μ from the body of rule (r_4) of Example 1 to $\mathcal{I}$ is given below:


μ
$\mathit{HPhone}(p_1, f) \wedge \mathit{HPhone}(p_2, f)$
$\mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}, \texttt{635}\})$
$\mathit{HPhone}([\mathsf{Doe}_3], \{\texttt{358}\})$

□

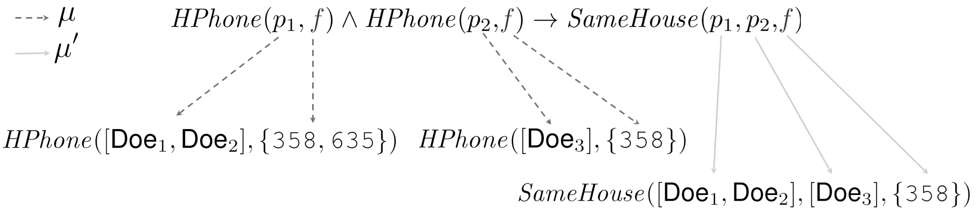
We are now ready to define the semantics of tgds and egds.

***Semantics of tgds* (Definition 4 of [3]).** An instance $\mathcal{I}$ for an ontology $\mathcal{O}$ satisfies a *tgd* $\forall \boldsymbol{x}(\phi(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \psi(\boldsymbol{x}, \boldsymbol{y}))$ if for each assignment μ from $\phi(\boldsymbol{x})$ to $\mathcal{I}$ there is an assignment μ' from

$\psi(\boldsymbol{x}, \boldsymbol{y})$ to $\mathcal{I}$ such that, for each x in $\boldsymbol{x}$

- $\mu(x) = \mu'(x)$, if x is an entity-variable
- the intersection of V-sets assigned by μ to each occurrence of x in $\phi(\boldsymbol{x})$ is contained in the intersection of V-sets assigned by μ' to each occurrence of x in $\psi(\boldsymbol{x}, \boldsymbol{y})$, if x is a value-variable

Example 5. Let $\mathcal{I}$ be the instance in Example 3 and (r_4) be the *tgd* in Example 1. Let μ be an assignment from the body of (r_4) to $\mathcal{I}$ and let μ' be an assignment from the head of (r_4) to $\mathcal{I}$ as described below:

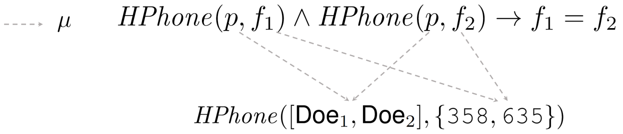

μ
μ'
$HPhone(p_1, f) \wedge HPhone(p_2,f) \rightarrow SameHouse(p_1, p_2,f)$
$HPhone([\mathsf{Doe}_1, \mathsf{Doe}_2], \{358, 635\})$
$HPhone([\mathsf{Doe}_3], \{358\})$
$SameHouse([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3], \{358\})$

Since μ is the only assignment from the body of (r_4) to $\mathcal{I}$, we conclude that $\mathcal{I}$ satisfies (r_4). □

Semantics of egds (Definition 4 of [3]). An instance $\mathcal{I}$ for an ontology $\mathcal{O}$ satisfies an *egd* $\forall \boldsymbol{x}(\phi(\boldsymbol{x}) \rightarrow y = z)$ if each assignment μ from $\phi(\boldsymbol{x})$ to $\mathcal{I}$ is such that $\mu(y) = \mu(z)$

Note that according to the above definition, in every assignment $\phi(\boldsymbol{x})$ to $\mathcal{I}$ all occurrences of y and z must be mapped to the same set, for an egd to be satisfied by an ontology instance.

Example 6. Let $\mathcal{I}$ be the instance in Example 3 and (r_3) be the *egd* in Example 1. Let μ be an assignment from the body of (r_3) to $\mathcal{I}$ as described below:


μ
$HPhone(p, f_1) \wedge HPhone(p, f_2) \rightarrow f_1 = f_2$
$HPhone([\mathsf{Doe}_1, \mathsf{Doe}_2], \{358, 635\})$

The assignment μ trivially enjoys the conditions of the definition of egd satisfaction. It is easy to see that the same holds for all assignments from the body of (r_3) to $\mathcal{I}$, and thus we conclude that $\mathcal{I}$ satisfies (r_3). □

Finally, we define when an instance is a solution for an ontology.

Ontology Solution **(Definition 5 of [3]).** Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology and $\mathcal{I}$ be an instance. $\mathcal{I}$ is a *solution* for $\mathcal{O}$ if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathcal{A}$, i.e.:

- $\mathcal{I}$ *satisfies* $\mathcal{T}$ if $\mathcal{I}$ satisfies every *tgd* and *egd* in $\mathcal{T}$.
- $\mathcal{I}$ satisfies $\mathcal{A}$ if it satisfies every ground atom $P(c_1, \ldots, c_n)$ in $\mathcal{A}$. A ground atom is satisfied if there is a fact $P(T_1, \ldots, T_n)$ in $\mathcal{I}$ such that $c_i \in T_i$, for $1 \leq i \leq n$.

Among all ontology solutions we are interested to the so-called *universal solutions*, which exhibit special properties with respect to the task of query answering (see, e.g., [5, 14, 15]).

Universal Solution **(Definitions 8 of [3]).** A solution $\mathcal{U}$ for an ontology $\mathcal{O}$ is *universal* if, for every solution $\mathcal{I}$ for $\mathcal{O}$, there is a homomorphism $h : \mathcal{U} \rightarrow \mathcal{I}$.

The notion of homomorphism used in the previous definition is tailored to our framework and suitably considers E-sets and V-sets occurring in instances. Informally, for an instance $\mathcal{I}$, we denote with $under(\mathcal{I})$ the set containing all entities, entity-nulls, values, and value-nulls occurring in $\mathcal{I}$. Then, let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two instances, an *homomorphism* $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ is a structure-preserving mapping from $under(\mathcal{I}_1)$ to $under(\mathcal{I}_2)$ (we refer to Definition 7 of [3] for further details).

3. Query Answering

A conjunctive query (CQ) $q(\boldsymbol{x})$ is a formula $\exists \boldsymbol{y}\, \phi(\boldsymbol{x}, \boldsymbol{y})$, where $\phi(\boldsymbol{x}, \boldsymbol{y})$ is a built-in free conjunction of atoms.

Answers to CQs. The *answer* $q^{\mathcal{I}}$ *to a CQ* $q(x_1, \ldots, x_n)$ *on an instance* $\mathcal{I}$ is the set of all tuples $\langle T_1, \ldots, T_n \rangle$ such that there is an assignment μ from q to $\mathcal{I}$ for which:

- $T_i = \mu(x_i)$, if x_i is an entity-variable;
- T_i, is the intersection of the V-sets assigned to the various occurrences of x_i, if x_i is a value-variable.

Example 7. Below we give an example of query, assignment μ from the query to the instance $\mathcal{I}$ of Example 3, and answers to the query.

$$\begin{tikzcd}
q(x) : -\mathit{HPhone}(\mathsf{Doe}_1, x) \arrow[d, dashed] & \wedge\ \mathit{HPhone}(\mathsf{Doe}_3, x) \arrow[d, dashed] \\
\mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{358, 635\}) & \mathit{HPhone}([\mathsf{Doe}_3], \{358\})
\end{tikzcd}$$

$$\dashrightarrow \ \mu$$

It is easy to see that $q^{\mathcal{I}} = \{\langle\{358\}\rangle\}$. □

When querying an ontology $\mathcal{O}$, we are interested in reasoning over all solutions for $\mathcal{O}$. We thus adapt the classical notion of certain answers to our framework. To this aim, we need some preliminary definitions. A tuple $\boldsymbol{T} = \langle T_1, \ldots, T_n \rangle$ is *null-free* if each T_i is non-empty and contains no nulls. Let $\boldsymbol{T} = \langle T_1, \ldots, T_n \rangle$ and $\boldsymbol{T'} = \langle T'_1, \ldots, T'_n \rangle$ be two tuples of E-sets and V-sets, we say that $\boldsymbol{T'}$ *dominates* $\boldsymbol{T}$, denoted $\boldsymbol{T} \leq \boldsymbol{T'}$, if $T_i \subseteq T'_i$, for all $1 \leq i \leq n$.

Certain Answers **(Definition 9 of [3]).** A null-free tuple $\boldsymbol{T}$ of E-sets and V-sets is a *certain answer* to a CQ q *w.r.t.* an ontology $\mathcal{O}$ if

1. for every solution $\mathcal{I}$ for $\mathcal{O}$, there is a tuple $\boldsymbol{T'} \in q^{\mathcal{I}}$ such that $\boldsymbol{T} \leq \boldsymbol{T'}$
2. there is no null-free tuple $\boldsymbol{T'}$ that satisfies (1) and $\boldsymbol{T} < \boldsymbol{T'}$

We write $cert(q, \mathcal{O})$ to denote the set of certain answers to q *w.r.t.* $\mathcal{O}$

We are now able to present the main result of this section, which asserts that universal solutions can be used to compute certain answers to CQs in our framework.

Compute Certain Answers over a Universal Solution **(Theorem 1 of [3])**. Let q be a CQ, let $\mathcal{O}$ be an ontology, and let $\mathcal{U}$ be a universal solution for $\mathcal{O}$. Then $cert(q, \mathcal{O}) = q^{\mathcal{U}}{\downarrow^{\rho}}$.

The operator $\downarrow^{\rho}$ first eliminates nulls occurring in $q^{\mathcal{U}}$, then, in the resulting set, it eliminates tuples that are dominated by other tuples (see [3] for the details).

4. Computing Universal Solution

In order to compute the universal solution that can be used to obtain the certain answers to CQs, we adapt here the well-known notion of restricted chase [4, 16, 5, 15] to our framework. Intuitively, given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, we define a procedure that starts from an instance for $\mathcal{O}$ "specular" to the ABox $\mathcal{A}$, which we call the base instance of $\mathcal{O}$, and incrementally constructs an instance that satisfies the tgds and the egds of the TBox $\mathcal{T}$.

Example 8. Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology such that $\mathcal{T}$ and $\mathcal{A}$ are as in Example 1 and in Example 2, respectively. The base instance of $\mathcal{O}$ is as follows:

$$\begin{array}{ll} (d_1)\ \mathit{Name}([\mathsf{Doe}_1], \{\texttt{John Doe}\}) & (d_2)\ \mathit{Name}([\mathsf{Doe}_2], \{\texttt{Johnny Doe}\}) \\ (d_3)\ \mathit{HPhone}([\mathsf{Doe}_1], \{\texttt{358}\}) & (d_4)\ \mathit{HPhone}([\mathsf{Doe}_2], \{\texttt{635}\}) \\ (d_5)\ \mathit{Name}([\mathsf{Doe}_3], \{\texttt{Mary Doe}\}) & (d_6)\ \mathit{HPhone}([\mathsf{Doe}_3], \{\texttt{358}\}) \end{array}$$ □

A universal solution for $\mathcal{O}$ is obtained by iteratively applying three chase rules, one for each kind of rule that may occur in $\mathcal{T}$, as long as they are triggered in the (current) instance.
(i) entity-egd chase rule. An entity-egd $\forall \boldsymbol{x}(\phi(\boldsymbol{x}) \rightarrow y = z)$ is triggered in an instance $\mathcal{I}$ if there is an assignment μ from $\phi(\boldsymbol{x})$ to $\mathcal{I}$ such that $\mu(y) \neq \mu(z)$. Then, the chase substitutes $\mu(y)$ and $\mu(z)$ everywhere $\mathcal{I}$ with $\mu(y) \cup \mu(z)$.

Example 9. Let $\mathcal{I}_0$ be the starting instance in Example 8 and (r_1) be the e-egd $Name(p_1, n_1) \wedge Name(p_2, n_2) \wedge JaccSim(n_1, n_2, 0.7) \rightarrow p_1 = p_2$. The egd (r_1) is triggered in $\mathcal{I}_0$ because of (d_1),(d_2). Its application generates $\mathcal{I}_1$, which is as follows:

$$\begin{array}{ll} (d_7)\ \mathit{Name}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{John Doe}\}) & (d_8)\ \mathit{Name}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{Johnny Doe}\}) \\ (d_9)\ \mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}\}) & (d_{10})\ \mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{635}\}) \\ (d_5)\ \mathit{Name}([\mathsf{Doe}_3], \{\texttt{Mary Doe}\}) & (d_6)\ \mathit{HPhone}([\mathsf{Doe}_3], \{\texttt{358}\}) \end{array}$$

$\mathcal{I}_1$ is obtained from $\mathcal{I}_0$ by replacing everywhere $[\mathsf{Doe}_1]$ and $[\mathsf{Doe}_2]$ with $[\mathsf{Doe}_1, \mathsf{Doe}_2]$. □

(ii) value-egd chase rule. Similarly to the entity-edg case, when a value-egd forces two different sets of values to be equated, we take the union of the two sets and modify the instance accordingly.

Example 10. Let $\mathcal{I}_1$ be the instance in Example 9 and (r_3) be the value-egd $HPhone(p, f_1) \wedge HPhone(p, f_2) \rightarrow f_1 = f_2$. The egd (r_3) is triggered in $\mathcal{I}_1$ because of (d_9),(d_10). Its application generates $\mathcal{I}_2$, shown below:

$$\begin{array}{ll} (d_7)\ \mathit{Name}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{John Doe}\}) & (d_8)\ \mathit{Name}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{Johnny Doe}\}) \\ (d_{11})\ \mathit{HPhone}([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}, \texttt{635}\}) & (d_6)\ \mathit{HPhone}([\mathsf{Doe}_3], \{\texttt{358}\}) \\ (d_5)\ \mathit{Name}([\mathsf{Doe}_3], \{\texttt{Mary Doe}\}) & \end{array}$$

$\mathcal{I}_2$ is obtained from $\mathcal{I}_1$ by replacing in d_9 and d_{10} $\{\texttt{358}\}$ and $\{\texttt{635}\}$ with their union $\{\texttt{358}, \texttt{635}\}$. □

(iii) tgd chase rule. A tgd $\forall \boldsymbol{x}\ (\phi(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y} \psi(\boldsymbol{x}, \boldsymbol{y}))$ is triggered in an instance $\mathcal{I}$ if there is an assignment μ from $\phi(\boldsymbol{x})$ to $\mathcal{I}$ such that there is no assignment μ' from $\psi(\boldsymbol{x}, \boldsymbol{y})$ to $\mathcal{I}$, as required by the definition of tgd satisfaction. Then, the chase adds new facts to $\mathcal{I}$ so that μ' does exist in the obtained instance.

Example 11. Let $\mathcal{I}_2$ be the instance in Example 10 and (r_4) be the tgd $HPhone(p_1, f) \wedge HPhone(p_2, f) \rightarrow SameHouse(p_1, p_2, f)$. The tgd (r_4) is triggered in $\mathcal{I}_2$ because of (d_{11}),(d_6). Its application generates $\mathcal{I}_3$, shown below:

(d_7) $Name([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{John Doe}\})$ (d_8) $Name([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{Johnny Doe}\})$
(d_{11}) $HPhone([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}, \texttt{635}\})$ (d_6) $HPhone([\mathsf{Doe}_3], \{\texttt{358}\})$
(d_5) $Name([\mathsf{Doe}_3], \{\texttt{Mary Doe}\})$ (d_{12}) $SameHouse([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3], \{\texttt{358}\})$

$\mathcal{I}_3$ is obtained by adding $SameHouse([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3], \{\texttt{358}\})$ to $\mathcal{I}_2$. □

The chase procedure terminates when it produces an instance for which no rule in the TBox is applicable. One such sequence $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_k$ is called a finite chase and the result of the chase, $chase(\mathcal{O}, \sigma)$, coincides with $\mathcal{I}_k$.

The result of the chase is an Universal Solution **(Theorem 2 of [3]).** If $\mathcal{O}$ is an ontology and σ is a finite chase for $\mathcal{O}$, then $chase(\mathcal{O}, \sigma)$ is a universal solution for $\mathcal{O}$.

Given the ontology composed by the TBox in Example 1 and the ABox in Example 2, it is possible to construct a finite chase sequence $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \mathcal{I}_k$, where $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ are as in Examples 8–11 and $\mathcal{I}_k$ is the universal solution of Example 3 (and thus $q^{\mathcal{I}}$ coincides with the set of certain answers to q).

5. Conclusion and Future Works

In this paper we reported on a principled approach to query answering and entity resolution that may be relevant to several areas, like Data Exchange [5] and Integration [17], or Ontology-based Data Access [18, 19] considered the central role played by tgds and egds in those contexts, but also in Information Extraction [20, 21, 22] or in general for data-intensive tasks [23]. Our investigation is however still initial and several aspects need further study. Although not explicitly shown in this document, tgds could give rise to an infinite universal solution because of the possible infinite creation of existentials. We left open how to obtain a universal solution from a non-terminating chase sequence. Solving this case would extend our approach even to settings where all universal solutions are infinite. Though not fully materializable, the infinite chase is generally considered an important theoretical tool to show properties of query answering, such as rewritability [15], possibly combined with partial materialization of the chase result [24]. Morover our framework intentionally considers expressive tgds and egds, in order to encompass several popular ontology languages. It is however well-known that without suitable restrictions query answering (and reasoning in general) is undecidable [25, 26]. It is thus crucial to identify conditions ensuring decidability, and possibly tractability, of query answering.

Acknowledgments

Scafoglieri's research was entirely and exclusively supported by PNRR MUR project PE0000013-FAIR. Lembo's research was supported by EU ICT-48 2020 project TAILOR (No. 952215), EU ERA-NET Cofund ICT-AGRI-FOOD project ADCATER (No. 40705), PNRR MUR project PE0000013-FAIR and the MUR PRIN 2022LA8XBH project Polar (POLicy specificAtion and enfoRcement for privacy-enhanced data management).

References

[1] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, J. Widom, Swoosh: a generic approach to entity resolution, Very Large Database J. 18 (2009) 255–276.

[2] G. Papadakis, E. Ioannou, E. Thanos, T. Palpanas, The Four Generations of Entity Resolution, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2021.

[3] R. Fagin, P. G. Kolaitis, D. Lembo, L. Popa, F. Scafoglieri, A Framework for Combining Entity Resolution and Query Answering in Knowledge Bases, in: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, 2023, pp. 229–239.

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

[5] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: Semantics and query answering, Theoretical Computer Science 336 (2005) 89–124.

[6] J. Baget, M. Leclère, M. Mugnier, E. Salvat, On rules with existential variables: Walking the decidability line, Artificial Intelligence 175 (2011) 1620–1654.

[7] M. Krötzsch, M. Marx, S. Rudolph, The power of the terminating chase (invited talk), in: Proc. of the 22nd Int. Conf. on Database Theory (ICDT), volume 127 of *LIPIcs*, 2019, pp. 3:1–3:17.

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, 2nd ed., Cambridge University Press, 2007.

[9] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, J. of Web Semantics 14 (2012) 57–83.

[10] M. Bienvenu, G. Cima, V. Gutieérrez-Basulto, LACE: A logical approach to collective entity resolution, in: Proc. of the 41st ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS), 2022, pp. 379–391.

[11] L. E. Bertossi, S. Kolahi, L. V. S. Lakshmanan, Data cleaning and query answering with matching dependencies and matching functions, Theoretical Computer Science 52 (2013) 441–482.

[12] W. Fan, Dependencies revisited for improving data quality, in: Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), 2008, pp. 159–170.

[13] K. Devlin, Sets, functions, and logic: an introduction to abstract mathematics, Chapman and Hall/CRC, 2018.

[14] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, J. of Artificial Intelligence Research 48 (2013) 115–174.

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

[16] D. S. Johnson, A. C. Klug, Testing containment of conjunctive queries under functional and inclusion dependencies, J. of Computer and System Sciences 28 (1984) 167–189.

[17] A. Doan, A. Y. Halevy, Z. G. Ives, Principles of Data Integration, Morgan Kaufmann, 2012.

[18] M. Bienvenu, M. Ortiz, Ontology-mediated query answering with data-tractable description

logics, in: W. Faber, A. Paschke (Eds.), Reasoning Web. Semantic Technologies for Intelligent Data Access – 11th Int. Summer School Tutorial Lectures (RW), volume 9203, 2015, pp. 218–307.

[19] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Ontology-based data access and integration, in: L. Liu, M. T. Özsu (Eds.), Encyclopedia of Database Systems, Second Edition, Springer, 2018.

[20] D. Lembo, Y. Li, L. Popa, K. Qian, F. Scafoglieri, Ontology mediated information extraction with MASTRO SYSTEM-T, in: Proceedings of the ISWC 2020 Demos and Industry Tracks, volume 2721 of *CEUR Electronic Workshop Proceedings,* http://ceur-ws.org/, CEUR-WS.org, 2020, pp. 256–261.

[21] D. Lembo, F. M. Scafoglieri, Ontology-based document spanning systems for information extraction, Int. J. Semantic Comput. 14 (2020) 3–26. URL: https://doi.org/10.1142/S1793351X20400012. doi:10.1142/S1793351X20400012.

[22] R. Fagin, B. Kimelfeld, F. Reiss, S. Vansummeren, Document spanners: A formal approach to information extraction, Journal of the ACM (JACM) 62 (2015) 1–51.

[23] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, K. Stefanidis, An overview of end-to-end entity resolution for big data, ACM Computing Surveys (CSUR) 53 (2020) 1–42.

[24] C. Lutz, D. Toman, F. Wolter, Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system, in: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI), 2009, pp. 2070–2075.

[25] C. Beeri, M. Y. Vardi, The implication problem for data dependencies, in: Proc. of the 8th Coll. on Automata, Languages and Programming (ICALP), volume 115 of *Lecture Notes in Computer Science*, Springer, 1981, pp. 73–85.

[26] A. Calì, D. Lembo, R. Rosati, On the decidability and complexity of query answering over inconsistent and incomplete databases, in: Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), 2003, pp. 260–271.