Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

A Framework for Combining Entity Resolution and Query Answering in Knowledge Bases

Ronald Fagin[1] , **Phokion G. Kolaitis**[1,2] , **Domenico Lembo**[3] , **Lucian Popa**[1] , **Federico Scafoglieri**[3]
[1]IBM Research, Almaden, USA
[2]UC Santa Cruz, USA
[3]Sapienza University of Rome, Italy
fagin@us.ibm.com, kolaitis@ucsc.edu, lembo@diag.uniroma1.it,
lpopa@us.ibm.com, scafoglieri@diag.uniroma1.it

Abstract

We propose a new framework for combining entity resolution and query answering in knowledge bases (KBs) with tuple-generating dependencies (tgds) and equality-generating dependencies (egds) as rules. We define the semantics of the KB in terms of special instances that involve equivalence classes of entities and sets of values. Intuitively, the former collect all entities denoting the same real-world object, while the latter collect all alternative values for an attribute. This approach allows us to both resolve entities and bypass possible inconsistencies in the data. We then design a chase procedure that is tailored to this new framework and has the feature that it never fails; moreover, when the chase procedure terminates, it produces a universal solution, which in turn can be used to obtain the certain answers to conjunctive queries. We finally discuss challenges arising when the chase does not terminate.

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, and so on (Benjelloun et al. 2009; Papadakis et al. 2021). In this paper, we study entity resolution in combination with query answering in the context of knowledge bases (KBs) 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 (Beeri and Vardi 1984; Fagin et al. 2005; Baget et al. 2011; Cuenca Grau et al. 2013; Krötzsch, Marx, and Rudolph 2019); in particular, they can express axioms that are used in Description Logic (DL) (Baader et al. 2007), as well as in specifying ontologies and KBs that are similar to Datalog +/- programs (Calì, Gottlob, and Lukasiewicz 2012). In addition, egds are employed to express typical entity resolution rules that one may write in practice, i.e., rules that enforce equality between two entities, as in (Bienvenu, Cima, and Gutiérrez-Basulto 2022).

The KBs considered here involve n-ary predicates that denote n-ary relations, $n \geq 1$, over *entities* and/or *values* from predefined datatypes. As is customary for ontologies, the *TBox* is the intensional component (i.e., the rules) of a KB, while the *database* of the KB is its extensional component (i.e., the ground atoms), sometimes called ABox.

As an example, Figure 1a depicts a set of ground atoms, where Doe_1 and Doe_2 are entities while the rest are values (indicating names and landline home phone numbers). Figure 2 illustrates a small TBox (containing only egds, for simplicity). To capture entity resolution rules, we allow egds to contain atoms involving built-in predicates, such as $JaccSim$. In the example, rule s_1 states that two names (i.e., strings) with Jaccard similarity above 0.6 must belong to the same individual. Rules s_2 and s_3 stipulate that an individual has at most one name and at most one landline home phone number, respectively. We call *entity-egds* rules that impose equality on two entities (e.g., s_1), and we call *value-egds* rules that impose equality on two values (e.g., s_2 and s_3).

It is now easy to see that, according to the standard semantics, the database in Figure 1a does not satisfy the entity-egd s_1 in Figure 2 (note that the Jaccard similarity of `John Doe` and `J. Doe` is 0.625). Thus, the main challenge one has in practice is to come up with a consistent way to complete or modify the original KB, while respecting all its rules.

In this paper, we develop a framework for entity resolution and query answering in KBs, where the valid models, called *KB solutions*, must satisfy all entity resolution rules along with all other KB 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, which 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, thus collecting together all possible values for a given attribute (e.g., the second argument of $Name$ or $HPhone$). Continuing with the above example, Figure 1b shows a new set of ground atoms that uses equivalence classes and sets of values.

We remark that the use of equivalence classes of entities and sets of values in KB solutions requires a drastic revision of the classical notion of satisfaction for tgds and egds. Intuitively, we interpret egds as matching dependencies (Bertossi, Kolahi, and Lakshmanan 2013; Fan 2008). That is, when the conditions expressed by (the body of) an entity-egd or a value-egd hold in the data (and thus two en-

(a) Before entity resolution.

(b) After entity resolution.

Figure 1: Ground Atoms of the Knowledge Base

$$
\begin{array}{ll}
(s_1) & \mathit{Name}(p_1, n_1) \wedge \mathit{Name}(p_2, n_2) \wedge \mathit{JaccSim}(n_1, n_2, 0.6) \\
& \rightarrow p_1 = p_2 \\
(s_2) & \mathit{Name}(p, n_1) \wedge \mathit{Name}(p, n_2) \rightarrow n_1 = n_2 \\
(s_3) & \mathit{HPhone}(p, f_1) \wedge \mathit{HPhone}(p, f_2) \rightarrow f_1 = f_2
\end{array}
$$

Figure 2: Entity-egds and value-egds.

tities or two values must be made equal), we combine them through a merging function. We adopt a general and common merging function that takes the union of the entities or the union of the values. This function actually belongs to the *Union class of match and merge functions* analyzed in (Benjelloun et al. 2009; Bertossi, Kolahi, and Lakshmanan 2013). In this way, we group all possible alternatives for denoting an object into a unique set of entities. 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.

It is worth noting that the semantics we adopt for value-egds allows us to always have a solution (that is, a model) for a KB, even when there are no models according to the standard first-order semantics; for example, if a value-egd enforces the equality of two different values, then first-order logic concludes that the KB is inconsistent. We thus may say that our semantics is inconsistency tolerant with respect to violations of value-egds. Indeed, we collect together all possible alternative values for individual attributes, whereas in data cleaning the task is to choose one of the alternatives.

We also point out that value-egds may substantially affect entity resolution. As an example, consider the database in Figure 1a and on top of it the new set of rules given below

$$
\begin{array}{ll}
(s_1') & \mathit{Name}(p_1, n_1) \wedge \mathit{HPhone}(p_1, f) \wedge \mathit{Name}(p_2, n_2) \wedge \\
& \mathit{HPhone}(p_2, f) \wedge \mathit{JaccSim}(n_1, n_2, 0.6) \rightarrow p_1 = p_2 \\
(s_2') & \mathit{Name}(p_1, n_1) \wedge \mathit{Name}(p_2, n_2) \wedge \mathit{HPhone}(p_1, f_1) \wedge \\
& \mathit{HPhone}(p_2, f_2) \wedge \mathit{JaccSim}(n_1, n_2, 0.6) \rightarrow f_1 = f_2.
\end{array}
$$

Rule s_1' states that two entities having similar names and the same landline home phone number denote the same real-world individual, while rule s_2' says that two entities with similar names must have the same phone number. It is easy to see that a TBox having only rule s_1' would not lead to inferring that Doe_1 and Doe_2 denote the same individual. However, by virtue of rule s_2', a solution has to group together the two telephone numbers into the same set, thus saying that Doe_1 and Doe_2 have both the landline home phone numbers $\{\mathtt{635}, \mathtt{358}\}$. This union "fires" rule s_1', and thus entities are resolved, i.e., are put together into the same equivalence class of the solution.

We finally note that, in order to maximize entity resolution, our semantics allows the body of a rule to be satisfied by assignments in which different occurrences of the same variable are replaced by different sets of values, as long as these sets have a non-empty intersection (instead of requiring that all occurrences are replaced by the same set of values). For instance, if we add to rules in Figure 2 the tgd

$$\mathit{HPhone}(p_1, f) \wedge \mathit{HPhone}(p_2, f) \rightarrow \mathit{SameHouse}(p_1, p_2, f),$$

stating that two entities with the same home phone number live in the same house, and to the database in Figure 1a the atom $\mathit{HPhone}(\mathsf{Doe}_3, \mathtt{358})$, we can conclude that the individual denoted by Doe_3 and the individual denoted by Doe_1 and Doe_2 live in the same house (since they share one phone number). Consistently with this choice, we also require tgds to "propagate" such intersections. Then, through the above tgd, we also infer that the phone number of the house is $\{\mathtt{358}\}$, which we denote with a fact of the form $\mathit{SameHouse}([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3], \{\mathtt{358}\})$. The behaviour we described differs from the standard one only for variables ranging on values, since two equivalence classes of entities are always either disjoint or the same class.

In this paper, we formalize the aforementioned ideas and investigate query answering, with focus on conjunctive queries (CQs). The main contributions are as follows.

- We propose a new framework for entity resolution in knowledge bases consisting of tgds and egds, and give rigorous semantics.
- 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 CQs.
- We propose a variant of the classical *chase* procedure (Beeri and Vardi 1984; Fagin et al. 2005) tailored to our approach. An important feature of our chase procedure is that it never fails, even in the presence of egds. At the same time, as in other frameworks, our chase procedure might not terminate.
- 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 conjunctive queries).

- When the chase procedure does not terminate, defining the result of the chase is more intricate. We show that the strategy proposed in (Beeri and Vardi 1984) does not work in our framework. Thus a different notion of the result of the chase is needed, which we leave for future study.

Detailed proofs are given in the arXiv version of this article: https://arxiv.org/abs/2303.07469.

2 Basic Notions

We take for granted the notions of equivalence relation and equivalence class. If $\mathcal{Z}$ is a set, θ is an equivalence relation on $\mathcal{Z}$, and $x \in \mathcal{Z}$, then the *equivalence class of x w.r.t. θ* is denoted by $[x]_\theta$ (or simply $[x]$, if θ is clear from the context). Sometimes, we write, e.g., $[a, b, c]_\theta$ (or $[a, b, c]$) to denote the equivalence class consisting of the elements a, b, and c. The *quotient set $\mathcal{Z}/\theta$ of θ on $\mathcal{Z}$* is the set of all equivalence classes over $\mathcal{Z}$ w.r.t. θ.

We consider four pairwise disjoint alphabets $\mathcal{S}_P$, $\mathcal{S}_E$, $\mathcal{S}_V$, and $\mathcal{S}_\mathcal{V}$. The set $\mathcal{S}_P$ is a finite alphabet for predicates; it is partitioned into the sets $\mathcal{S}_O$ and $\mathcal{S}_B$, which are the alphabets for *KB* predicates and *built-in* predicates, respectively. The sets $\mathcal{S}_E$, $\mathcal{S}_V$, and $\mathcal{S}_\mathcal{V}$ are countable infinite alphabets for *entities*, *values*, and *variables*, respectively. For ease of exposition, we do not distinguish between different data types, thus $\mathcal{S}_V$ is a single set containing all possible values.

The number of arguments of a predicate $P \in \mathcal{S}_P$ is the *arity* of P, denoted with $arity(P)$. With each n-ary predicate P we associate a tuple $type(P) = \langle \rho_1, \ldots \rho_n \rangle$, such that, for each $1 \leq i \leq n$, either $\rho_i = \mathsf{e}$ or $\rho_i = \mathsf{v}$. This tuple specifies the *types of the arguments of* P, i.e., whether each argument of P ranges over entities (e) or values (v). We also write $type(P, i)$ for the *type* ρ_i of the i-th argument of P.

Note that the built-in predicates from $\mathcal{S}_B$ are special, pre-interpreted predicates, whose arguments range only over values, i.e., if B is an n-ary built-in predicate, then $type(B)$ is the n-ary tuple $\langle \mathsf{v}, \ldots, \mathsf{v} \rangle$. The examples in Sec. 1 use the Jaccard similarity $JaccSim$ as a built-in predicate.

An *atom* is an expression $P(t_1, \ldots, t_n)$, where $P \in \mathcal{S}_P$, and each t_i is a term, i.e., is either a variable from $\mathcal{S}_\mathcal{V}$ or a constant, which in turn is either an entity from $\mathcal{S}_E$, if $type(P, i) = \mathsf{e}$, or a value from $\mathcal{S}_V$, if $type(P, i) = \mathsf{v}$. When $t_i \in \mathcal{S}_\mathcal{V}$, we call it an *entity-variable* if $type(P, i) = \mathsf{e}$, or a *value-variable*, if $type(P, i) = \mathsf{v}$. A *ground* atom is an atom with no variables.

A *conjunction $\phi(\mathbf{x})$ of atoms* is an expression $P_1(\mathbf{t}_1) \wedge \ldots \wedge P_m(\mathbf{t}_m)$, where each $P_j(\mathbf{t}_j)$ is an atom such that each variable in the tuple $\mathbf{t}_j$ of terms is among those in the tuple $\mathbf{x}$ of variables. We also require that every variable x in $\mathbf{x}$ is either an entity-variable or a value-variable and that, if x occurs in an atom whose predicate is built-in (note that thus x is a value-variable), then there exists some other atom $P_j(\mathbf{t}_j)$ in $\phi(\mathbf{x})$ such that $P_j \in \mathcal{S}_O$ and x is in $\mathbf{t}_j$. If a conjunction contains no built-in predicates, we say that it is *built-in free*; if it contains no entities or values, we say that it is *constant-free*.

As done in this section, in the formulas appearing throughout the paper, we use e to denote an entity from $\mathcal{S}_E$, we use v to denote a value from $\mathcal{S}_V$, we use c to denote a constant (i.e., $c \in \mathcal{S}_E \cup \mathcal{S}_V$), we use x, y, w, and z to denote variables from $\mathcal{S}_\mathcal{V}$, and we use t to denote terms (i.e., $t \in \mathcal{S}_\mathcal{V} \cup \mathcal{S}_E \cup \mathcal{S}_V$). Typically, we use P to denote a predicate from $\mathcal{S}_P$. All the above symbols may appear with subscripts. Moreover, we use bold font for tuples of variables, terms, and so on. In the examples, we use self-explanatory symbols; we write entities in Helvetica font and values in `true type` font.

3 Framework

In this section, we present the syntax and the semantics of a framework for entity resolution in knowledge bases.

Syntax. A *knowledge base* (KB) $\mathcal{K}$ is a pair $(\mathcal{T}, \mathbf{D})$, consisting of a TBox $\mathcal{T}$ and a database $\mathbf{D}$. The TBox is a finite set of *tuple-generating dependencies* (tgds) and *equality-generating dependencies* (egds). A *tgd* is a formula

$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})), \tag{1}$$

where $\phi(\mathbf{x})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atoms, such that $\mathbf{x}$ and $\mathbf{y}$ have no variables in common and $\psi(\mathbf{x}, \mathbf{y})$ is built-in free and, for simplicity, constant-free. As in (Fagin et al. 2005), we assume that all variables in $\mathbf{x}$ appear in $\phi(\mathbf{x})$, but not necessarily in $\psi(\mathbf{x}, \mathbf{y})$. We call $\phi(\mathbf{x})$ the body of the tgd, and $\psi(\mathbf{x}, \mathbf{y})$ the head of it.

An *egd* is a formula

$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow y = z), \tag{2}$$

where $\phi(\mathbf{x})$ is a conjunction of atoms and y and z are distinct variables occurring in $\phi(\mathbf{x})$, 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 $\phi(\mathbf{x})$ the body of the egd. For value-egds, we require that neither y nor z occur in atoms having a built-in as predicate. This ensures that the meaning of built-ins remains fixed. We will write $\mathsf{body}(r)$ to denote the body of a tgd or an egd r; furthermore, we may write r with no quantifiers.

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

(r_1) $CI(p_1, name_1, phone_1) \wedge CI(p_2, name_2, phone_2) \wedge JaccSim(name_1, name_2, 0.6) \rightarrow p_1 = p_2$
(r_2) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow name_1 = name_2$
(r_3) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow phone_1 = phone_2$
(r_4) $CI(p, name, phone) \rightarrow Emp(p, comp) \wedge CEO(comp, dir)$
(r_5) $Emp(p, comp_1) \wedge Emp(p, comp_2) \rightarrow comp_1 = comp_2$
(r_6) $CI(p_1, name_1, phone) \wedge CI(p_2, name_2, phone) \rightarrow SameHouse(p_1, p_2, phone)$.

Here, $type(CI) = \langle \mathsf{e}, \mathsf{v}, \mathsf{v} \rangle$, $type(Emp) = \langle \mathsf{e}, \mathsf{e} \rangle$, $type(CEO) = \langle \mathsf{e}, \mathsf{e} \rangle$, and $type(SameHouse) = \langle \mathsf{e}, \mathsf{e}, \mathsf{v} \rangle$. The predicates have the meaning suggested by their names. In particular, the predicate CI maintains contact information of individuals, whereas Emp associates employees to the companies they work for. Each of the six rules makes an assertion about the predicates. In particular, rule r_1 states that if the Jaccard similarity of two names is higher then 0.6, then these names are names of the same individual. $\square$

The database $\mathbf{D}$ of a KB $\mathcal{K}$ is a finite set of ground atoms of the form $P(c_1, \ldots, c_n)$ over the alphabets $\mathcal{S}_P$, $\mathcal{S}_E$, and $\mathcal{S}_V$, where $P \in \mathcal{S}_P$ and each c_i is an entity from $\mathcal{S}_E$, if $type(P, i) = \mathsf{e}$, or a value from $\mathcal{S}_V$, if $type(P, i) = \mathsf{v}$.

Example 2. Let $\mathbf{D}$ be the database consisting of the atoms

(g_1) $CI(\mathsf{Doe}_1, \texttt{J. Doe}, \texttt{358})$, (g_4) $Emp(\mathsf{Doe}_2, \mathsf{Yahoo})$
(g_2) $CI(\mathsf{Doe}_2, \texttt{John Doe}, \texttt{635})$, (g_5) $Emp(\mathsf{Doe}_3, \mathsf{IBM})$
(g_3) $CI(\mathsf{Doe}_3, \texttt{Mary Doe}, \texttt{358})$, (g_6) $CEO(\mathsf{Yahoo}, \mathsf{Doe}_1)$.

In words, the database $\mathbf{D}$ specifies that Doe_1 has name `J. Doe` and phone number `358` (g_1), Doe_2 has name `John Doe` and phone number `635` (g_2), Doe_3 has name `Mary Doe` and phone number `358` (g_3), Doe_2 is employee of Yahoo (g_4), Doe_3 is employee of IBM (g_5), and the CEO of Yahoo is Doe_1 (g_6). □

To ensure that built-in predicates have the same semantics in every KB, we assume that we have a fixed (infinite and countable) set GB of ground atoms of the form $B(v_1, ..v_n)$, where B is in $\mathcal{S}_B$ and $v_1, ..v_n$ are in $\mathcal{S}_V$. Intuitively, GB contains all facts about built-in predicates that hold overall. Given a KB $\mathcal{K} = (\mathcal{T}, \mathbf{D})$, we assume that $\mathbf{D} = \mathbf{D}_O \cup \mathbf{D}_B$, where $\mathbf{D}_O$ contains only ground atoms with predicate from $\mathcal{S}_O$ and $\mathbf{D}_B$ is the (finite) set of all atoms in GB whose built-in predicates and values are mentioned in $\mathcal{T}$ and in $\mathbf{D}_O$.

Semantics. Let $\mathcal{S}_{EN}$ and $\mathcal{S}_{VN}$ be two infinite, countable, disjoint sets that are also disjoint from the alphabets introduced in Sec. 2. We call $\mathcal{S}_{EN}$ the set of entity-nulls and $\mathcal{S}_{VN}$ the set of value-nulls; their union is referred to as the set of nulls. We use $sig(\mathcal{K})$ to denote the signature of a KB $\mathcal{K}$, i.e., the set of symbols of $\mathcal{S}_P$, $\mathcal{S}_E$ and $\mathcal{S}_V$ occurring in $\mathcal{K}$.

The semantics of a KB is given using special databases, called KB *instances*, whose ground atoms have components that are either equivalence classes of entities and entity-nulls or non-empty sets of values and value-nulls.

Definition 1. Let $\mathcal{K}$ be a KB, $\mathcal{S}$ a subset of $(\mathcal{S}_E \cap sig(\mathcal{K})) \cup \mathcal{S}_{EN}$, and $\sim$ an equivalence relation on $\mathcal{S}$. An *instance* $\mathcal{I}$ *for* $\mathcal{K}$ *w.r.t.* $\sim$ is a set of *facts* $P(T_1, \ldots, T_n)$ such that $P \in \mathcal{S}_P \cap sig(\mathcal{K})$, $arity(P) = n$, and, for each $1 \leq i \leq n$, we have that $T_i \neq \emptyset$ and either $T_i \in \mathcal{S}/{\sim}$, if $type(P, i) = \mathsf{e}$, or $T_i \subseteq (\mathcal{S}_V \cap sig(\mathcal{K})) \cup \mathcal{S}_{VN}$, if $type(P, i) = \mathsf{v}$. The relation $\sim$ is called the equivalence relation *associated* with $\mathcal{I}$.

To denote an equivalence class in $\mathcal{S}/{\sim}$, we may use the symbol E; also, to denote a non-empty subset of $(\mathcal{S}_V \cap sig(\mathcal{K})) \cup \mathcal{S}_{VN}$, we may use the symbol V. Even though E may contains nulls, we will often call E simply equivalence class of entities. Similarly, V will be simply called set of values. We will then use T to denote a set that is either E or V. All such symbols may occur with a subscript.

Definition 2. Let $\mathcal{K}$ be a KB and let $\mathcal{I}$ be an instance for $\mathcal{K}$ w.r.t. to an equivalence relation $\sim$.

The *active domain of* $\mathcal{I}$, denoted by $active(\mathcal{I})$, is the set $\{T \mid \text{there are } P(T_1, \ldots, T_n) \in \mathcal{I} \text{ and } i \leq n \text{ with } T = T_i\}$.

We write $active_E(\mathcal{I})$ and $active_V(\mathcal{I})$ to denote the set of all equivalence classes of entities and the set of all sets of values contained in $active(\mathcal{I})$, respectively. Obviously, $active(\mathcal{I}) = active_E(\mathcal{I}) \cup active_V(\mathcal{I})$.

The *underlying domain of* $\mathcal{I}$, denoted by $under(\mathcal{I})$, is the set $under_E(\mathcal{I}) \cup under_V(\mathcal{I})$, where

$under_E(\mathcal{I}) = \{e \mid \text{there is } E \in active_E(\mathcal{I}) \text{ and } e \in E\}$
and
$under_V(\mathcal{I}) = \{v \mid \text{there is } V \in active_V(\mathcal{I}) \text{ and } v \in V\}$.

Note that an instance $\mathcal{I}$ of $\mathcal{K}$ w.r.t. $\sim$ is also an instance of $\mathcal{K}$ w.r.t. the equivalence relation $\sim'$ induced on $under_E(\mathcal{I})$ by $\sim$. Therefore, in what follows, we will consider only instances w.r.t. equivalence relations over $under_E(\mathcal{I})$.

Furthermore, we may simply call $\mathcal{I}$ an instance for $\mathcal{K}$ and leave the equivalence relation associated with $\mathcal{I}$ implicit.

Example 3. Let $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ be a KB such that $\mathcal{T}$ and $\mathbf{D}$ are as in Example 1 and in Example 2, respectively. Then, consider the following set $\mathcal{I}$ of facts:

(d_1) $CI([\mathsf{Doe}_1], \{\texttt{J. Doe}\}, \{\texttt{358}\})$
(d_2) $CI([\mathsf{Doe}_2], \{\texttt{John Doe}\}, \{\texttt{635}\})$
(d_3) $CI([\mathsf{Doe}_3], \{\texttt{Mary Doe}\}, \{\texttt{358}\})$
(d_4) $Emp([\mathsf{Doe}_2], [\mathsf{Yahoo}])$
(d_5) $Emp([\mathsf{Doe}_3], [\mathsf{IBM}])$
(d_6) $CEO([\mathsf{Yahoo}], [\mathsf{Doe}_1])$.

$\mathcal{I}$ is an instance for $\mathcal{K}$ w.r.t. the identity relation over the set $\mathcal{S} = \{\mathsf{Doe}_1, \mathsf{Doe}_2, \mathsf{Doe}_3, \mathsf{Yahoo}, \mathsf{IBM}\}$. Further, let $\mathsf{e}_1^\perp$ and $\mathsf{e}_2^\perp$ be entity-nulls, and $\sim'$ be an equivalence relation over $\mathcal{S} \cup \{\mathsf{e}_1^\perp, \mathsf{e}_2^\perp\}$, such that $\mathsf{Doe}_1 \sim' \mathsf{Doe}_2$, $\mathsf{IBM} \sim' \mathsf{e}_1^\perp$ and their symmetric versions are the only equivalences in $\sim'$ different from the identity. The following set $\mathcal{I}'$ of facts is an instance for $\mathcal{K}$ w.r.t. $\sim'$.

$CI([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{J. Doe}, \texttt{John Doe}\}, \{\texttt{358}, \texttt{635}\})$
$CI([\mathsf{Doe}_3], \{\texttt{Mary Doe}\}, \{\texttt{358}\})$
$Emp([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Yahoo}])$
$Emp([\mathsf{Doe}_3], [\mathsf{IBM}, \mathsf{e}_1^\perp])$
$CEO([\mathsf{Yahoo}], [\mathsf{Doe}_1, \mathsf{Doe}_2])$
$CEO([\mathsf{IBM}, \mathsf{e}_1^\perp], [\mathsf{e}_2^\perp])$
$SameHouse([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Doe}_3], \{\texttt{358}\})$
$SameHouse([\mathsf{Doe}_3], [\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{358}\})$.

□

To define the notions of satisfaction of tgds and egds by an instance, we first introduce the notion of an *assignment* from a conjunction $\phi(\mathbf{x})$ of atoms to an instance $\mathcal{I}$ of a KB $\mathcal{K}$. To formalize this notion, we need a preliminary transformation τ of $\phi(\mathbf{x})$ that substitutes each occurrence of a value-variable in $\phi(\mathbf{x})$ with a fresh variable. We call such fresh variables *set-variables* and denote the result of the transformation $\tau(\phi(\mathbf{x}))$. If x is a value-variable in $\mathbf{x}$, we write $SetVar(x, \tau(\phi(\mathbf{x})))$ to denote the set of fresh variables used in $\tau(\phi(\mathbf{x}))$ to replace the occurrences of x in $\phi(\mathbf{x})$. For example, if $\phi(\mathbf{x}) = P_1(x, y, z) \wedge P_2(y, z) \wedge P_3(x, w)$, where $type(P_1) = \langle \mathsf{e}, \mathsf{e}, \mathsf{v} \rangle$, $type(P_2) = \langle \mathsf{e}, \mathsf{v} \rangle$, and $type(P_3) = \langle \mathsf{e}, \mathsf{v} \rangle$, then $\tau(\phi(\mathbf{x})) = P_1(x, y, S_1^z) \wedge P_2(y, S_2^z) \wedge P_3(x, S_1^w)$, where S_1^z, S_2^z, S_1^w are the fresh set-variables introduced by τ. Also, $SetVar(z, \tau(\phi(\mathbf{x}))) = \{S_1^z, S_2^z\}$ and $SetVar(w, \tau(\phi(\mathbf{x}))) = \{S_1^w\}$. In what follows, if x is a value-variable, then each set-variable in $SetVar(x, \tau(\phi(\mathbf{x})))$ will have x as a superscript.

We are now ready to formally define assignments.

Definition 3. Let $\phi(\mathbf{x})$ be a conjunction of atoms and let $\mathcal{I}$ be an instance for a KB $\mathcal{K}$ w.r.t. $\sim$. An *assignment from* $\phi(\mathbf{x})$ *to* $\mathcal{I}$ is a mapping μ from the variables and values in $\tau(\phi(\mathbf{x}))$ to $active(\mathcal{I})$, defined as follows:

1. $\mu(x)$ is an equivalence class in $active_E(\mathcal{I})$, for every entity-variable x;
2. $\mu(v) = V$ such that $V \in active_V(\mathcal{I})$ and $v \in V$, for every value v;
3. $\mu(S)$ is a set of values in $active_V(\mathcal{I})$, for every set-variable S;
4. $\bigcap_{i=1}^{k} \mu(S_i^x) \neq \emptyset$, for every value-variable x such that $SetVar(x, \tau(\phi(\mathbf{x}))) = \{S_1^x, \ldots, S_k^x\}$;
5. $\mathcal{I}$ contains a fact of the form $P(\mu(x), \mu(S), [e]_\sim, \mu(v))$, for each atom $P(x, S, e, v)$ of $\tau(\phi(\mathbf{x}))$, where x is an entity-variable, S is a set-variable, e is an entity in $\mathcal{S}_E$ and v is a value in $\mathcal{S}_V$ (the definition generalizes in the obvious way for atoms of different form).

In words, the above definition says that an assignment maps every entity-variable to an equivalence class of entities (Condition 1), every value to a set of values containing it (Condition 2), and every occurrence of a value-variable to a set of values (Condition 3), in such a way that multiple occurrences of the same value-variable are mapped to sets with a non-empty intersection (Condition 4). This captures the intuition that, since predicate arguments ranging over values are interpreted through sets of values, a join between such arguments holds when such sets have a non-empty intersection. Finally, Condition 5 states that $\tau(\phi(\mathbf{x}))$ is "realized" in $\mathcal{I}$. Note that an assignment also maps values to sets of values (Condition 2). This allows an assignment to map atoms in $\phi(\mathbf{x})$ to facts in $\mathcal{I}$, since predicate arguments ranging over values are instantiated in $\mathcal{I}$ by sets of values.

If $P(\mathbf{t})$ is an atom of $\phi(\mathbf{x})$, we write $\mu(P(\mathbf{t}))$ for the fact of $\mathcal{I}$ in Condition 5 and call it the *μ-image (in $\mathcal{I}$) of $P(\mathbf{t})$*. We write $\mu(\phi(\mathbf{x}))$ for the set $\{\mu(P(\mathbf{t})) \mid P(\mathbf{t}) \text{ occurs in } \phi(\mathbf{x})\}$, and call it the *$\mu$-image (in $\mathcal{I}$) of $\phi(\mathbf{x})$*.

Example 4. Consider the tgd r_6 of Example 1 and the instance $\mathcal{I}'$ given in Example 3 and apply τ to the body of r_6 to obtain $CI(p_1, S^{name_1}, S_1^{phone}) \wedge CI(p_2, S^{name_2}, S_2^{phone})$. Let μ be the following mapping:

$$\mu(p_1) = [\mathsf{Doe}_1, \mathsf{Doe}_2],\ \mu(S^{name_1}) = \{\texttt{J. Doe}, \texttt{John Doe}\}$$
$$\mu(S_1^{phone}) = \{\texttt{358}, \texttt{635}\},\ \mu(p_2) = [\mathsf{Doe}_3]$$
$$\mu(S^{name_2}) = \{\texttt{Mary Doe}\},\ \mu(S_2^{phone}) = \{\texttt{358}\}.$$

It is easy to see that μ is an assignment from the body of r_6 to $\mathcal{I}'$ (note the non-empty intersection between $\mu(S_1^{phone})$ and $\mu(S_2^{phone})$). Let us now apply τ to the head of r_6 to obtain $SameHouse(p_1, p_2, R^{phone})$. The following mapping μ' is an assignment from $SameHouse(p_1, p_2, phone)$ to $\mathcal{I}'$:

$$\mu'(p_1) = [\mathsf{Doe}_1, \mathsf{Doe}_2], \mu'(p_2) = [\mathsf{Doe}_3], \mu'(R^{phone}) = \{\texttt{358}\}.$$

□

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

Definition 4. An instance $\mathcal{I}$ for a KB $\mathcal{K}$ satisfies:

- a tgd of the form (1), if for each assignment μ from $\phi(\mathbf{x})$ to $\mathcal{I}$ there is an assignment μ' from $\psi(\mathbf{x}, \mathbf{y})$ to $\mathcal{I}$ such that, for each x in $\mathbf{x}$ occurring in both $\phi(\mathbf{x})$ and $\psi(\mathbf{x}, \mathbf{y})$:
  - $\mu(x) = \mu'(x)$, if x is an entity-variable;
  - $\bigcap_{i=1}^{k} \mu(S_i^x) \subseteq \bigcap_{i=1}^{\ell} \mu'(R_i^x)$, where $\{S_1^x, \ldots, S_k^x\} = SetVar(x, \tau(\phi(\mathbf{x})))$ and $\{R_1^x, \ldots, R_\ell^x\} = SetVar(x, \tau(\psi(\mathbf{x}, \mathbf{y})))$, if x is a value-variable.

  Each such assignment μ' is called a *head-compatible tgd-extension* of μ to $\mathcal{I}$ (or simply a tgd-extension of μ to $\mathcal{I}$).
- an entity-egd of the form (2), if each assignment μ from $\phi(\mathbf{x})$ to $\mathcal{I}$ is such that $\mu(y) = \mu(z)$;
- a value-egd of the form (2), if each assignment μ from $\phi(\mathbf{x})$ to $\mathcal{I}$ is such that $\mu(S^y) = \mu(S^z)$ for all set-variables $S^y, S^z \in SetVar(y, \tau(\phi(\mathbf{x}))) \cup SetVar(z, \tau(\phi(\mathbf{x})))$.

In words, Definition 4 expresses the following for each assignment μ from the body of a rule r to $\mathcal{I}$:

(1) If r is a tgd, the definition stipulates two different behaviours for frontier variables (i.e., variables occurring in both the body and the head of the tgd). Specifically, for the tgd to be satisfied, it should be possible to extend μ so that (i) every frontier entity-variable is assigned to exactly the same equivalence class both in the body and the head of the rule (this is in line with the standard semantics for frontier variables in tgds); and (ii) the intersection of the sets of values assigned in the body of the tgd to the various occurrences of a frontier value-variable x is contained in the intersection of the sets of values assigned to the occurrences of x in the head of the tgd. Note that, while μ maps all the occurrences of an entity-variable in $\phi(\mathbf{x})$ to the same equivalence class, it may map multiple occurrences of a value-variable to different sets of values with non empty-intersection; thus, it is natural that this intersection is "propagated" to the head.

(2) If r is an entity-egd, the notion of satisfaction is standard, since it requires that μ assigns y and z to the same equivalence class of entities.

(3) If r is a value-egd, we have again to take into account that each occurrence of y in the rule body can be assigned by μ to a different set of values, and analogously for each occurrence of z. The definition stipulates that, for the value-egd to be satisfied, all such sets of values are equal.

Example 5. $\mathcal{I}'$ from Example 3 satisfies tgd r_6 of Example 1, since μ' is a head-compatible tgd extension of the only assignment μ from the body of r_6 to $\mathcal{I}'$. Indeed, $\mu'(p_1) = \mu(p_1)$, $\mu'(p_2) = \mu(p_2)$, $\mu'(R^{phone}) = \mu(S_1^{phone}) \cap \mu(S_2^{phone})$ (cf. Example 4). It is also easy to see that $\mathcal{I}'$ satisfies all rules of Example 1 (note that we left implicit the facts over built-in predicates of $\mathcal{I}'$). □

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

Definition 5. Let $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ be a KB and $\mathcal{I}$ an instance.

- $\mathcal{I}$ *satisfies* $\mathcal{T}$ if $\mathcal{I}$ satisfies every tgd and egd in $\mathcal{T}$.
- $\mathcal{I}$ satisfies a ground atom $P(c_1, \ldots, c_n)$ in $\mathbf{D}$ if there is a fact $P(T_1, \ldots, T_n)$ in $\mathcal{I}$ such that $c_i \in T_i$, for $1 \leq i \leq n$.
- $\mathcal{I}$ satisfies $\mathbf{D}$ if $\mathcal{I}$ satisfies all ground atoms in $\mathbf{D}$.
- $\mathcal{I}$ is a *solution* for $\mathcal{K}$, denoted by $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ satisfies $\mathcal{T}$ and $\mathbf{D}$. The set of all solutions of a KB $\mathcal{K}$ is denoted by $Sol(\mathcal{K})$, i.e., $Sol(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \models \mathcal{K}\}$.

Example 6. Consider the instances $\mathcal{I}$ and $\mathcal{I}'$ given in Example 3 for the $\mathcal{K}$ of Example 1 and Example 2. It is easy to see that $\mathcal{I}$ is not a solution for $\mathcal{K}$, whereas $\mathcal{I}'$ is a solution for $\mathcal{K}$. □

Universal solutions. It is well known that the *universal* solutions exhibit good properties that make them to be the preferred solutions (see, e.g., (Fagin et al. 2005; Calì, Gottlob, and Kifer 2013; Calvanese et al. 2007)). To introduce the notion of a universal solution in our framework, we first need to adapt the notion of homomorphism. We begin with some auxiliary definitions and notation.

Definition 6. Let $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ and $\mathbf{T}' = \langle T'_1, \ldots, T'_n\rangle$ be two tuples such that each T_i and each T'_i is either a set of entities and entity-nulls or a set of values and value-nulls.

- $\mathbf{T}'$ *dominates* $\mathbf{T}$, denoted $\mathbf{T} \leq \mathbf{T}'$, if $T_i \subseteq T'_i$, for all i.
- $\mathbf{T}'$ *strictly dominates* $\mathbf{T}$, denoted $\mathbf{T} < \mathbf{T}'$, if $\mathbf{T} \leq \mathbf{T}'$ and $\mathbf{T} \neq \mathbf{T}'$.
- Let $P(\mathbf{T})$ and $P(\mathbf{T}')$ be facts. $P(\mathbf{T}')$ *dominates* $P(\mathbf{T})$, denoted $P(\mathbf{T}) \leq P(\mathbf{T}')$, if $\mathbf{T} \leq \mathbf{T}'$; $P(\mathbf{T}')$ *strictly dominates* $P(\mathbf{T})$, denoted $P(\mathbf{T}) < P(\mathbf{T}')$, if $\mathbf{T} < \mathbf{T}'$.

Definition 7. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two instances of a KB $\mathcal{K}$. A *homomorphism* $h : \mathcal{I}_1 \to \mathcal{I}_2$ is a mapping from the elements of $under(\mathcal{I}_1)$ to elements of $under(\mathcal{I}_2)$ such that:

1. $h(e) = e$, for every entity e in $under_E(\mathcal{I}_1) \cap \mathcal{S}_E$;
2. $h(e_\perp)$ belongs to $under_E(\mathcal{I}_2)$, for every entity-null $e_\perp$ in $under_E(\mathcal{I}_1) \cap \mathcal{S}_{EN}$;
3. $h(v) = v$, for every value v in $under_V(\mathcal{I}_1) \cap \mathcal{S}_V$;
4. $h(v_\perp)$ belongs to $under_V(\mathcal{I}_2)$, for every value-null $v_\perp$ in $under_V(\mathcal{I}_1) \cap \mathcal{S}_{VN}$;
5. for every $P(T_1, \ldots, T_n)$ in $\mathcal{I}_1$, there is a $P(U_1, \ldots, U_n)$ in $\mathcal{I}_2$ such that $P(h(T_1), \ldots, h(T_n)) \leq P(U_1, \ldots, U_n)$, where $h(T_i) = \{h(x) \mid x \in T_i\}$, for $1 \leq i \leq n$.

In the sequel, we may use $h(\langle T_1, \ldots, T_n\rangle)$ to denote the tuple $\langle h(T_1), \ldots, h(T_n)\rangle$.

We now define the key notion of a universal solution.

Definition 8. A solution $\mathcal{U}$ for a KB $\mathcal{K}$ is *universal* if, for every $\mathcal{I} \in Sol(\mathcal{K})$, there is a homomorphism $h : \mathcal{U} \to \mathcal{I}$.

The instance $\mathcal{I}'$ in Example 3 is a universal solution for $\mathcal{K}$, as is the instance obtained by eliminating $\mathsf{e}_1^\perp$ from $\mathcal{I}'$.

Two instances $\mathcal{I}_1$ and $\mathcal{I}_2$ are *homomorphically equivalent* if there are homomorphisms $h : \mathcal{I}_1 \to \mathcal{I}_2$ and $h' : \mathcal{I}_2 \to \mathcal{I}_1$. All universal solutions are homomorphically equivalent.

4 Query Answering

A conjunctive query (CQ) q is a formula $\exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{y})$ is a built-in free conjunction of atoms. The *arity* of q is the number of its free variables in $\mathbf{x}$; we will often write $q(\mathbf{x})$, instead of just q, to indicate the free variables of q.

Let $q(\mathbf{x}) : \exists \mathbf{x} \phi(\mathbf{x}, \mathbf{y})$ be a CQ, where $\mathbf{x} = x_1, \ldots, x_n$. Given a KB $\mathcal{K}$ and an instance $\mathcal{I}$ for $\mathcal{K}$, the *answer to q on $\mathcal{I}$*, denoted by $q^{\mathcal{I}}$, is the set of all tuples $\langle T_1, \ldots, T_n\rangle$ such that there is an assignment μ from $\phi(\mathbf{x}, \mathbf{y})$ to $\mathcal{I}$ for which

- $T_i = \mu(x_i)$, if x_i is an entity-variable;
- $T_i = \bigcap_{j=1}^{k} \mu(S_j^{x_i})$, if x_i is a value-variable, where $\{S_1^{x_i}, \ldots, S_k^{x_i}\} = SetVar(x_i, \tau(\phi(\mathbf{x}, \mathbf{y})))$.

We will also say that μ is an assignment from $q(\mathbf{x})$ to $\mathcal{I}$.

Example 7. Let $q(x)$ be the CQ:

$$\exists p_1, p_2\; CI(p_1, \texttt{J. Doe}, x) \wedge CI(p_2, \texttt{Mary Doe}, x)$$

asking for the phone number in common between (the entities named) `J. Doe` and `Mary Doe`. The answer to $q(x)$ on the instance $\mathcal{I}'$ in Example 3 is $q^{\mathcal{I}'} = \{\langle\{\texttt{358}\}\rangle\}$. This is obtained through the assignment μ_q defined as follows:

$$\mu_q(p_1) = [\mathsf{Doe}_1, \mathsf{Doe}_2], \mu_q(\texttt{J. Doe}) = \{\texttt{J. Doe}, \texttt{John Doe}\}$$
$$\mu_q(S_1^x) = \{\texttt{358}, \texttt{635}\}, \mu_q(p_2) = [\mathsf{Doe}_3]$$
$$\mu_q(\texttt{Mary Doe}) = \{\texttt{Mary Doe}\}, \mu_q(S_2^x) = \{\texttt{358}\}.$$

If $q_1(x) : \exists z\; CEO(z, x)$ is the query asking for the CEOs, then $q_1^{\mathcal{I}'} = \{\langle[\mathsf{Doe}_1, \mathsf{Doe}_2]\rangle, \langle[\mathsf{e}_2^\perp]\rangle\}$. □

The next result tells how conjunctive queries are preserved under homomorphisms in our framework.

Proposition 1. Let q be a CQ and let $\mathcal{K}$ a KB. If $\mathcal{I}_1, \mathcal{I}_2$ are two instances for $\mathcal{K}$ and $h : \mathcal{I}_1 \to \mathcal{I}_2$ is a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$, then, for every $\mathbf{T} \in q^{\mathcal{I}_1}$, there is $\mathbf{U} \in q^{\mathcal{I}_2}$ such that $h(\mathbf{T}) \leq \mathbf{U}$.

When querying a KB $\mathcal{K}$, we are interested in reasoning over all solutions for $\mathcal{K}$. We adapt the classical notion of certain answers to our framework. A tuple $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ is *null-free* if each T_i is non-empty and contains no nulls.

Definition 9. Let q be a CQ and let $\mathcal{K}$ be a KB. A null-free tuple $\mathbf{T}$ is a *certain answer* to q w.r.t. $\mathcal{K}$ if

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

We write $cert(q, \mathcal{K})$ for the set of certain answers to q.

Note that the second condition in the above definition asserts that a certain answer has to be a maximal null-free tuple with respect to the tuple dominance order $\leq$ given in Definition 6, among the tuples satisfying Condition 1.

The next result tells that if two sets of entities appear in a certain answer or in different certain answers, then either they are the same set or they are disjoint. In particular, the sets of entities that appear in certain answers can be viewed as equivalence classes of some equivalence relation.

Proposition 2. Let q be a CQ of arity n, let $\mathcal{K}$ be a KB, and let $\mathbf{T} = \langle T_1, \ldots, T_n\rangle$ and $\mathbf{T}' = \langle T'_1, \ldots, T'_n\rangle$ be two certain answers to q w.r.t. $\mathcal{K}$. If T_i and T'_j are sets of entities, then either $T_i = T'_j$ or $T_i \cap T'_j = \emptyset$, where $1 \leq i, j \leq n$.

In data exchange and related areas, the certain answers to CQs can be obtained by evaluating the query on a universal solution and then applying an operator $\downarrow$ that eliminates the tuples that contain nulls (see, e.g., (Fagin et al. 2005; Calvanese et al. 2007)). The operator $\downarrow$ can easily be adapted to our framework as follows. If E is a set of entities and entity-nulls, then $E{\downarrow} = \{e \mid e \in \mathcal{S}_E \cap E\}$. Similarly, if V is a non-empty set of values and value-nulls, then $V{\downarrow} = \{v \mid v \in \mathcal{S}_V \cap V\}$. In words, $\downarrow$ removes all nulls from E and from V. If $\langle T_1, \ldots, T_n\rangle$ is a tuple such that each T_i is either a set of entities and entity-nulls or a set of values and value-nulls, then we set $\langle T_1, \ldots, T_n\rangle{\downarrow} = \langle T_1{\downarrow}, \ldots, T_n{\downarrow}\rangle$. Finally, given a set Θ of tuples of the above form, $\Theta{\downarrow}$ is the set obtained from Θ by removing all tuples in Θ containing a T_i such that $T_i\downarrow = \emptyset$, and replacing every other tuple $\langle T_1, \dots, T_n\rangle$ in Θ by the tuple $\langle T_1, \dots, T_n\rangle\downarrow$.

The next example shows that, in our framework, the certain answers to a CQ q cannot always be obtained by evaluating q on a universal solution and then applying $\downarrow$.

Example 8. Let P_1 and P_2 be such that $type(P_1) = type(P_2) = \langle \mathsf{e}, \mathsf{v}\rangle$, $\mathcal{T}'$ be the TBox consisting of the rules

$$P_1(x, y) \to P_2(x, y),\ \ P_1(x, y) \wedge P_1(x, z) \to y = z$$

and $\mathbf{D}' = \{P_1(\mathsf{e}, 1), P_1(\mathsf{e}, 2)\}$. Consider the following universal solutions for $\mathcal{K}' = (\mathcal{T}', \mathbf{D}')$

$$\begin{array}{lcl}\mathcal{I}_1 & = & \{P_1([\mathsf{e}], \{\mathtt{1,2}\}), P_2([\mathsf{e}], \{\mathtt{1,2}\})\},\\ \mathcal{I}_2 & = & \{P_1([\mathsf{e}], \{\mathtt{1,2}\}), P_2([\mathsf{e}], \{\mathtt{1}\}), P_2([\mathsf{e}], \{\mathtt{1,2}\})\},\end{array}$$

and the query $q(x, y)\ :\ P_2(x, y)$. Then $q^{\mathcal{I}_1}\downarrow = q^{\mathcal{I}_1} = \{\langle[\mathsf{e}], \{\mathtt{1,2}\}\rangle\}$, and $q^{\mathcal{I}_2}\downarrow = q^{\mathcal{I}_2} = \{\langle[\mathsf{e}], \{\mathtt{1,2}\}\rangle, \langle[\mathsf{e}], \{\mathtt{1}\}\rangle\}$. Thus, $q^{\mathcal{I}_2}\downarrow$ does not coincide with the set of certain answers to q w.r.t. $\mathcal{K}'$, because the tuple $\langle[\mathsf{e}], \{\mathtt{1}\}\rangle$ does not satisfy the second condition in Definition 9. $\square$

Intuitively, in the above example the universal solution $\mathcal{I}_2$ is not "minimal", in the sense that the fact $P_2([\mathsf{e}], \{\mathtt{1}\})$ in $\mathcal{I}_2$ is dominated by another fact of $\mathcal{I}_2$, namely $P_2([\mathsf{e}], \{\mathtt{1,2}\})$. This behaviour causes that the answer to the query over $\mathcal{I}_2$ contains also tuples that are not maximal with respect to tuple dominance (cf. Definition 9). This suggests that we need some additional processing besides elimination of nulls. We modify the operator $\downarrow$ by performing a further *reduction* step.

Definition 10. Given a query q and an instance $\mathcal{I}$ for a KB $\mathcal{K}$, we write $q^{\mathcal{I}}\downarrow^{\rho}$ to denote the set of null-free tuples obtained by removing from $q^{\mathcal{I}}\downarrow$ all tuples strictly dominated by other tuples in the set, that is, if $\mathbf{T}$ and $\mathbf{T}'$ are two tuples in $q^{\mathcal{I}}\downarrow$ such that $\mathbf{T} < \mathbf{T}'$, then remove $\mathbf{T}$ from $q^{\mathcal{I}}\downarrow$.

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

Theorem 1. Let q be a CQ, let $\mathcal{K}$ be a KB, and let $\mathcal{U}$ be a universal solution for $\mathcal{K}$. Then $cert(q, \mathcal{K}) = q^{\mathcal{U}}\downarrow^{\rho}$.

Example 9. As seen earlier in Example 6, $\mathcal{I}'$ is a universal solution for $\mathcal{K}$. Then, for the query $q_1(x)\ :\ \exists z CEO(z, x)$, we have that $cert(q_1, \mathcal{K}) = q_1^{\mathcal{I}'}\downarrow^{\rho} = \{\langle[\mathsf{Doe}_1, \mathsf{Doe}_2]\rangle\}$. $\square$

5 Computing a Universal Model

In this section, we adapt the well-known notion of restricted chase (Beeri and Vardi 1984; Johnson and Klug 1984; Fagin et al. 2005; Calvanese et al. 2007) to our framework. Interestingly, the chase procedure we define never produces a failure, unlike, e.g., the restricted chase procedure in the case of standard data exchange, where the application of egds may cause a failure when two different constants have to be made equal. Instead, in our framework, when an entity-egd forces two different equivalence classes of entities to be equated, we combine the two equivalence classes into a bigger equivalence class. Similarly, 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. However, it is possible that the chase procedure may have infinitely many steps, each producing a new instance. As a consequence, some care is required in defining the result of the application of this (potentially infinite) procedure to a KB $\mathcal{K}$, so that we can obtain an instance that can be used (at least in principle) for query answering. We call such instance *the result of the chase of the KB* $\mathcal{K}$, and distinguish the case in which the chase terminates from the case in which it does not. In the former case, the result of the chase of $\mathcal{K}$ is simply the instance produced in the last step of the chase procedure, and we show that this instance is a universal solution for $\mathcal{K}$. In the latter case, we point out that previous approaches from the literature for infinite standard chase sequences under tgds and egds cannot be smoothly adapted to our framework, and we leave it open for this case how to define the result of the chase so that it is a universal solution.

We start with the notion of the *base instance* for a KB. Given a KB $\mathcal{K} = (\mathcal{T}, \mathbf{D})$, we define the set

$$\mathcal{I}^{\mathbf{D}} = \{P(\{c_1\}, \dots, \{c_n\}) \mid P(c_1, \dots, c_n) \in \mathbf{D}\}.$$

$\mathcal{I}^{\mathbf{D}}$ is an instance for $\mathcal{K}$ w.r.t. the identity relation id over the set $\mathcal{S}_E \cap sig(\mathcal{K})$. We call $\mathcal{I}^{\mathbf{D}}$ the *base instance* for $\mathcal{K}$. Note that the base instance for $\mathcal{K}$ is also a base instance for every KB having $\mathbf{D}$ as database, and is a solution for $(\emptyset, \mathbf{D})$. As an example, note that the instance $\mathcal{I}$ of Example 3 is the base instance for the KB $\mathcal{K}$ defined in Example 1 and in Example 2.

We next define three chase steps, one for tgds, one for entity-egds, and one for value-egds.

Definition 11. Let $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ be a KB and $\mathcal{I}_1$ an instance for $\mathcal{K}$ w.r.t. an equivalence relation $\sim^1$ on $under_E(\mathcal{I}_1)$.

- (tgd) Let r be a tgd of the form (1). Without loss of generality, we assume that all atoms in $\psi(\mathbf{x}, \mathbf{y})$ are of the form $P(x_1, \dots, x_k, y_1, \dots, y_\ell)$, where $x_1, \dots, x_k$ belong to $\mathbf{x}$ and $y_1, \dots, y_\ell$ belong to $\mathbf{y}$, and we denote with $\mathbf{y_e} = y_e^1, \dots y_e^h$ and $\mathbf{y_v} = y_v^1, \dots, y_v^j$ the entity-variables and value-variables in $\mathbf{y}$, respectively. Let μ be an assignment from $\phi(\mathbf{x})$ to $\mathcal{I}_1$ such that there is no $\psi(\mathbf{x}, \mathbf{y})$-compatible tgd-extension of μ to $\mathcal{I}_1$. We say that r is *applicable* to $\mathcal{I}_1$ with μ (or that μ *triggers* r *in* $\mathcal{I}_1$), and construct $\mathcal{I}_2$ via the following procedure:

  let $\{f_e^1, \dots, f_e^h\} \subseteq \mathcal{S}_{EN}$ and $\{f_v^1, \dots, f_v^j\} \subseteq \mathcal{S}_{VN}$ be two sets of fresh nulls (i.e., not occurring in $\mathcal{I}_1$), which are distinct from each other
  put $\mathcal{I}_2 := \mathcal{I}_1$
  for each atom $P(x_1, \dots, x_k, y_1, \dots, y_\ell)$ in $\psi(\mathbf{x}, \mathbf{y})$ **do**
  $\mathcal{I}_2 := \mathcal{I}_2 \cup \{P(T_1, \dots, T_k, U_1, \dots, U_\ell)\}$

  where,
  - for $1 \leq i \leq k$, we have $T_i = \mu(x_i)$, if x_i is an entity-variable, or $T_i = \bigcap_{p=1}^{m} \mu(S_p^{x_i})$, if x_i is a value-variable and $\{S_1^{x_i}, \dots, S_m^{x_i}\} = SetVar(x_i, \tau(\phi(\mathbf{x})))$;
  - for $1 \leq i \leq \ell$, we have $U_i = \{f_e^s\}$ if $y_i = y_e^s$, with $1 \leq s \leq h$, or $U_i = \{f_v^q\}$ if $y_i = y_v^q$, with $1 \leq q \leq j$. (Thus, each singleton $\{f_e^s\}$ is a new equivalence class.)

- (entity-egd) Let r be an entity-egd of the form (2), and μ an assignment from $\phi(\mathbf{x})$ to $\mathcal{I}_1$ such that $\mu(y) \neq \mu(z)$. We say that r is *applicable* to $\mathcal{I}_1$ with μ (or that μ *triggers* r in $\mathcal{I}_1$), and we construct $\mathcal{I}_2$ from $\mathcal{I}_1$ by replacing in $\mathcal{I}_1$ all occurrences of $\mu(y)$ and $\mu(z)$ with $\mu(y) \cup \mu(z)$. (Thus, we merge two equivalence classes into a new one.)
- (value-egd) let r be a value-egd of the form (2). Let $\{S_1^y, \ldots, S_m^y\} = SetVar(y, \tau(\phi(\mathbf{x})))$, $\{S_1^z, \ldots, S_k^z\} = SetVar(z, \tau(\phi(\mathbf{x})))$, $1 \leq i \leq m$, $1 \leq j \leq k$, and μ be an assignment from $\phi(\mathbf{x})$ to $\mathcal{I}_1$ such that $\mu(S_i^y) \neq \mu(S_j^z)$. We say that r is *applicable* to $\mathcal{I}_1$ with μ (or that μ *triggers* r in $\mathcal{I}_1$), and we construct $\mathcal{I}_2$ from $\mathcal{I}_1$ by replacing in the image $\mu(\phi(\mathbf{x}))$ each set $\mu(S_1^y), \ldots, \mu(S_m^y), \mu(S_1^z), \ldots$ and $\mu(S_k^z)$ with $\mu(S_1^y) \cup \ldots \cup \mu(S_m^y) \cup \mu(S_1^z) \cup \ldots \cup \mu(S_k^z)$.

If r is a tgd or egd that can be applied to $\mathcal{I}_1$ with μ, we say that $\mathcal{I}_2$ is the *result of applying* r to $\mathcal{I}_1$ with μ and we write $\mathcal{I}_1 \xrightarrow{r,\mu} \mathcal{I}_2$. We call $\mathcal{I}_1 \xrightarrow{r,\mu} \mathcal{I}_2$ a *chase step*.

For both the entity-egd step and the value-egd step, the chase procedure constructs $\mathcal{I}_2$ by replacing some facts of $\mathcal{I}_1$. However, whereas for entity-egds the replacement is "global" (i.e., the two equivalence classes merged in the step are substituted by their union everywhere in $\mathcal{I}_1$), for value-egds the replacement is "local", in the sense that the two sets merged in the step are substituted by their union *only* in facts occurring in the image $\mu(\phi(\mathbf{x}))$, which is a subset of $\mathcal{I}_1$.

Example 10. Consider again the KB $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ of Example 1 and Example 2. The instance $\mathcal{I}$ of Example 3 is the base instance $\mathcal{I}^{\mathbf{D}}$ for $\mathcal{K}$. We depict below the application of the rules of Definition 11, starting from the instance $\mathcal{I} = \mathcal{I}^{\mathbf{D}}$.

tgd application: $\mathcal{I}^{\mathbf{D}} \xrightarrow{r_4,\mu_0} \mathcal{I}_1$, where μ_0 is such that $\mu_0(\mathsf{body}(r_4)) = \{d_3\}$. The set $\mathcal{I}_1$ consists of the facts d_1-d_6 (see Example 3), as well as the facts:

$$(d_7) Emp([\mathsf{Doe}_3], [\mathsf{e}_1^{\perp}]), \ \ (d_8) CEO([\mathsf{e}_1^{\perp}], [\mathsf{e}_2^{\perp}]).$$

entity-egd application: $\mathcal{I}_1 \xrightarrow{r_1,\mu_1} \mathcal{I}_2$, where μ_1 is such that $\mu_1(\mathsf{body}(r_1)) = \{d_1, d_2, JaccSim(\{\texttt{J. Doe}\}, \{\texttt{John Doe}\}, \{0.6\})\}$. The set $\mathcal{I}_2$ consists of the facts d_3, d_5, d_7, d_8, as well as the facts:

$$\begin{array}{l}
(d_1) \rightarrow (d_9) \ \ CI([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{J. Doe}\}, \{\texttt{358}\}) \\
(d_2) \rightarrow (d_{10}) \ CI([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{John Doe}\}, \{\texttt{635}\}) \\
(d_4) \rightarrow (d_{11}) \ Emp([\mathsf{Doe}_1, \mathsf{Doe}_2], [\mathsf{Yahoo}]) \\
(d_6) \rightarrow (d_{12}) \ CEO([\mathsf{Yahoo}], [\mathsf{Doe}_1, \mathsf{Doe}_2]).
\end{array}$$

value-egd application: $\mathcal{I}_2 \xrightarrow{r_2,\mu_2} \mathcal{I}_3$, where μ_2 is such that $\mu_2(\mathsf{body}(r_2)) = \{d_9, d_{10}\}$. The set $\mathcal{I}_3$ consists of the facts d_3, d_{11}, d_5, d_{12}, d_7, d_8, as well as the facts:

$$\begin{array}{ll}
(d_9) \rightarrow (d_{13}) & CI([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{J. Doe}, \texttt{John Doe}\}, \\
& \{\texttt{358}\}) \\
(d_{10}) \rightarrow (d_{14}) & CI([\mathsf{Doe}_1, \mathsf{Doe}_2], \{\texttt{J. Doe}, \texttt{John Doe}\}, \\
& \{\texttt{635}\}).
\end{array}$$

□

We now define the notion of a chase sequence.

Definition 12. Let $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ be a KB. A *chase sequence for* $\mathcal{K}$ is a (finite or infinite) sequence $\sigma = \mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \ldots$, such that $\mathcal{I}_0 = \mathcal{I}^{\mathbf{D}}$ and $\mathcal{I}_i \xrightarrow{r_i,\mu_i} \mathcal{I}_{i+1}$ is a chase step, for each consecutive pair $(\mathcal{I}_i, \mathcal{I}_{i+1})$ in the sequence σ.

The following propositions state some basic properties of the elements in a chase sequence.

Proposition 3. Let $\mathcal{K}$ be a KB.

- If $\mathcal{I}$ is an instance for $\mathcal{K}$ w.r.t. an equivalence relation $\sim$ on $under_E(\mathcal{I})$ and if $\mathcal{I}'$ is such that $\mathcal{I} \xrightarrow{r,\mu} \mathcal{I}'$, then there is an equivalence relation $\sim'$ such that $\mathcal{I}'$ is an instance for $\mathcal{K}$ w.r.t. $\sim'$ on $under_E(\mathcal{I}')$.
- Every $\mathcal{I}_i$ occurring in a chase sequence for a KB $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ is an instance for $\mathcal{K}$.

Proposition 4. Let $\mathcal{K}$ be a KB, let σ be a chase sequence for $\mathcal{K}$, and let $\mathcal{I}_i$ and $\mathcal{I}_j$, with $i \leq j$, be two instances for $\mathcal{K}$ w.r.t. equivalence relations $\sim^i$ and $\sim^j$, respectively, such that $\mathcal{I}_i$ and $\mathcal{I}_j$ belong to σ. Then the following holds:

- $\sim^i \subseteq \sim^j$ and $under(\mathcal{I}_i) \subseteq under(\mathcal{I}_j)$;
- If $P(\mathbf{T})$ is a fact in $\mathcal{I}_i$, then there is a fact $P(\mathbf{T}')$ in $\mathcal{I}_j$ such that $P(\mathbf{T}) \leq P(\mathbf{T}')$.

Proposition 4 implies that for every $i \leq j$ and every equivalence class E in $under_E(\mathcal{I}_i)/\sim^i$, there exists E' in $under_E(\mathcal{I}_j)/\sim^j$ such that $E \subseteq E'$; in particular, we have that $[e]_{\sim^i} \subseteq [e]_{\sim^j}$, for each entity $e \in \mathcal{S}_E \cap sig(\mathcal{K})$.

When a chase sequence for a KB is infinite, there might be rules applicable to some instance $\mathcal{I}_i$ in the sequence that are never applied, even though they remain applicable in subsequent chase steps. It is however always possible to establish a suitable order in the application of the rules so that the above situation never occurs. Chase sequences enjoying this property are called *fair*. In what follows, we assume that all chase sequences considered are fair.

We now investigate the possible outcomes of the chase for a KB $\mathcal{K}$. We first give the definition of the result of the chase for a finite chase sequence. Then we attack the case of infinite sequences.

Finite chase sequences. In this case, we can provide the following natural definition.

Definition 13. Let $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ be a KB.

- A *finite chase of* $\mathcal{K}$ is a finite chase sequence $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_m$, for which there is no rule r in $\mathcal{T}$ and no assignment μ such that r can be applied to $\mathcal{I}_m$ with μ.
- We say that $\mathcal{I}_m$ is the *result of the finite chase* σ *for* $\mathcal{K}$, and denote it by $chase(\mathcal{K}, \sigma)$.

Proposition 3 implies that $chase(\mathcal{K}, \sigma)$ is an instance for $\mathcal{K}$.

Example 11. Consider the KB $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ of Example 1 and Example 2. As said, interpretation $\mathcal{I}$ of Example 3 is the base instance for $\mathcal{K}$, i.e., $\mathcal{I} = \mathcal{I}^{\mathbf{D}}$. Consider now, in the same order, the three chase steps described in Example 10, and continue the chase procedure from $\mathcal{I}_3$ as follows:

Step 4: $\mathcal{I}_3 \xrightarrow{r_3,\mu_4} \mathcal{I}_4$, where μ_3 is such that $\mu_3(\mathsf{body}(r_3)) = \{d_{13}, d_{14}\}$. The resulting instance $\mathcal{I}_4$ consists of the fact d_3,

d_{11}, d_5, d_{12}, d_7, d_8, as well as the fact:

$$(d_{13}),(d_{14}) \to (d_{15})\ \mathit{CI}([\mathsf{Doe}_1,\mathsf{Doe}_2],\{\texttt{J. Doe},\texttt{John Doe}\},\{\texttt{358},\texttt{635}\}).$$

Note that d_{15} replaces facts d_{13} and d_{14} of $\mathcal{I}_3$.

Step 5: $\mathcal{I}_4 \xrightarrow{r_5,\mu_5} \mathcal{I}_5$, where μ_5 is such that $\mu_5(\mathsf{body}(r_5)) = \{d_5, d_7\}$. The resulting instance $\mathcal{I}_5$ consists of the facts d_{14}, d_3, d_{11}, d_{12}, as well as the facts:

$$\begin{array}{ll} (d_5),(d_7) \to (d_{16}) & \mathit{Emp}([\mathsf{Doe}_3],[\mathsf{IBM},\mathsf{e}_1^{\perp}]) \\ (d_8) \to (d_{17}) & \mathit{CEO}([\mathsf{IBM},\mathsf{e}_1^{\perp}],[\mathsf{e}_2^{\perp}]). \end{array}$$

Note that d_{16} replaces facts d_5 and d_7 of $\mathcal{I}_4$.

Step 6: $\mathcal{I}_5 \xrightarrow{r_6,\mu_6} \mathcal{I}_6$, where μ_6 is such that $\mu_6(\mathsf{body}(r_6)) = \{d_{14}, d_3\}$. The resulting instance $\mathcal{I}_6$ consists of the same facts as $\mathcal{I}_5$, as well as the fact:

$$(d_{18})\quad \mathit{SameHouse}([\mathsf{Doe}_1,\mathsf{Doe}_2],[\mathsf{Doe}_3],\{\texttt{358}\}).$$

Step 7: $\mathcal{I}_6 \xrightarrow{r_6,\mu_7} \mathcal{I}_7$, where μ_7 is such that $\mu_7(\mathsf{body}(r_6)) = \{d_{14}, d_3\}$. The resulting instance $\mathcal{I}_7$ contains the same facts as $\mathcal{I}_6$, as well as the fact:

$$(d_{19})\quad \mathit{SameHouse}([\mathsf{Doe}_3],[\mathsf{Doe}_1,\mathsf{Doe}_2],\{\texttt{358}\}).$$

It is not difficult to see that no rule of $\mathcal{T}$ can be applied to $\mathcal{I}_7$, and thus the chase procedure terminates. Therefore, the sequence $\sigma = \mathcal{I}^{\mathbf{D}}, \mathcal{I}_1, \ldots, \mathcal{I}_7$ is a finite chase for $\mathcal{K}$, and $\mathit{chase}(\mathcal{K},\sigma) = \mathcal{I}_7$ (note that $\mathcal{I}_7$ coincides with $\mathcal{I}'$ given in Example 3). □

Actually, $\mathit{chase}(\mathcal{K},\sigma)$ turns out to be a solution for $\mathcal{K}$.

Lemma 1. Let $\mathcal{K}$ be a KB and let $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_m$ be a finite chase for $\mathcal{K}$. Then $\mathit{chase}(\mathcal{K},\sigma)$ is a solution for $\mathcal{K}$.

The following lemma is used to prove that if σ is a finite chase, then $\mathit{chase}(\mathcal{K},\sigma)$ is not just a solution for $\mathcal{K}$, but also a universal solution for $\mathcal{K}$. A similar result, known as the Triangle Lemma, was used in (Fagin et al. 2005) in the context of data exchange.

Lemma 2. Let $\mathcal{I}_1 \xrightarrow{r,\mu} \mathcal{I}_2$ be a chase step. Let $\mathcal{I}$ be an instance for $\mathcal{K}$ such that $\mathcal{I}$ satisfies r and there exists a homomorphism $h_1 : \mathcal{I}_1 \to \mathcal{I}$. Then there is a homomorphism $h_2 : \mathcal{I}_2 \to \mathcal{I}$ such that h_2 extends h_1.

The following theorem is the main result of this section.

Theorem 2. If $\mathcal{K}$ is a KB and $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_m$ is a finite chase for $\mathcal{K}$, then $\mathit{chase}(\mathcal{K},\sigma)$ is a universal solution for $\mathcal{K}$.

Theorem 2 implies that if both σ and σ' are finite chases for $\mathcal{K}$, then $\mathit{chase}(\mathcal{K},\sigma)$ and $\mathit{chase}(\mathcal{K},\sigma')$ are homomorphically equivalent. Thus, the result of a finite chase for $\mathcal{K}$ is unique up to homomorphic equivalence.

It is easy to verify that if $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ is a KB in which every tgd in $\mathcal{T}$ is full (i.e., there are no existential quantifiers in the heads of the tgds in $\mathcal{T}$), then every chase sequence for $\mathcal{K}$ is finite. In particular, this holds true if $\mathcal{T}$ consists of egds only, which covers all entity resolution settings.

Infinite chase sequences. For infinite chase sequences, the definition of the result of the chase requires some care. The typical approach adopted when only tgds are present (Krötzsch, Marx, and Rudolph 2019; Grahne and Onet 2018), or for settings involving *separable* tgds and egds (Johnson and Klug 1984; Calvanese et al. 2007; Calì, Gottlob, and Kifer 2013), is to define the result of an infinite chase sequence as the union of all facts generated in the various chase steps. This approach does not work for arbitrary tgds and egds since the resulting instance needs not satisfy all the rules and thus needs not be a solution in our terminology. Moreover, in our framework the union of all instances in an infinite chase sequence is not even an instance for the KB at hand, because the sets of entities in the result do not correspond to equivalence classes with respect to an equivalence relation.

An alternative approach, which might be better suited for a setting with arbitrary tgds and egds, is to define the result of an infinite chase sequence as the instance containing all *persistent* facts, i.e., all facts that are introduced in some step in the chase sequence and are never modified in subsequent chase steps. This notion was introduced in (Beeri and Vardi 1984). In our setting, given a KB $\mathcal{K}$ and an infinite chase sequence $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots$, this means that we define the result $\mathit{chase}(\mathcal{K},\sigma)$ of the infinite chase σ of $\mathcal{K}$ as follows:

$$\mathit{chase}(\mathcal{K},\sigma) = \{f \mid \text{there is some } i \geq 0 \text{ such that } f \in \mathcal{I}_j \text{ for each } j \geq i\}. \quad (3)$$

The following example shows that this definition does not work in our framework, because the above set might be empty (even if the database $\mathbf{D}$ is non-empty).

Example 12. Let $\mathcal{K} = \langle \mathcal{T}, \mathbf{D} \rangle$, where $\mathbf{D} = \{P(\texttt{1},\texttt{2})\}$ and $\mathcal{T}$ consists of the two rules

$$\begin{array}{ll} (r_1) & P(x,y) \to P(y,z) \\ (r_2) & P(x,y) \wedge P(y,z) \to y = z \end{array}$$

with $\mathit{type}(P) = \langle \mathsf{v}, \mathsf{v} \rangle$. We construct an infinite chase sequence starting with $\mathcal{I}_0 = \{P(\{\texttt{1}\},\{\texttt{2}\})\}$ and by repeatedly applying the above rules with suitable assignments in the following order: $r_1, r_1, r_2, r_1, r_2, r_1, r_2, r_1 \ldots$.

We obtain the infinite chase sequence $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots$

$$\begin{array}{lcl} \mathcal{I}_0 & = & \{P(\{\texttt{1}\},\{\texttt{2}\})\} \\ \mathcal{I}_1 & = & \{P(\{\texttt{1}\},\{\texttt{2}\}), P(\{\texttt{2}\},\{\mathrm{v}_1^{\perp}\})\} \\ \mathcal{I}_2 & = & \{P(\{\texttt{1}\},\{\texttt{2}\}), P(\{\texttt{2}\},\{\mathrm{v}_1^{\perp}\}), P(\{\mathrm{v}_1^{\perp}\},\{\mathrm{v}_2^{\perp}\})\} \\ \mathcal{I}_3 & = & \{P(\{\texttt{1}\},\{\texttt{2},\mathrm{v}_1^{\perp}\}), P(\{\texttt{2},\mathrm{v}_1^{\perp}\},\{\texttt{2},\mathrm{v}_1^{\perp}\}), \\ & & P(\{\mathrm{v}_1^{\perp}\},\{\mathrm{v}_2^{\perp}\})\} \\ \ldots & & \\ \mathcal{I}_7 & = & \{P(\{\texttt{1}\},\{\texttt{2},\mathrm{v}_1^{\perp},\mathrm{v}_2^{\perp}\}), P(\{\texttt{2},\mathrm{v}_1^{\perp}\},\{\texttt{2},\mathrm{v}_1^{\perp},\mathrm{v}_2^{\perp}\}), \\ & & P(\{\texttt{2},\mathrm{v}_1^{\perp},\mathrm{v}_2^{\perp}\},\{\texttt{2},\mathrm{v}_1^{\perp},\mathrm{v}_2^{\perp}\}), P(\{\mathrm{v}_2^{\perp}\},\{\mathrm{v}_3^{\perp}\}), \\ & & P(\{\mathrm{v}_3^{\perp}\},\{\mathrm{v}_4^{\perp}\})\} \\ \ldots & & \end{array}$$

It is not difficult to see that the set $\mathit{chase}(\mathcal{K},\sigma)$ defined in (3) above is empty, i.e., *no* persistent facts occur in σ. □

The infinite chase sequence σ in Example 12 is fair. At the same time, there are finite fair sequences for the KB $\mathcal{K}$ in this example. Indeed, if we apply rule r_1 and then rule r_2, we get a finite chase sequence $\sigma' = \mathcal{I}'_0, \mathcal{I}'_1, \mathcal{I}'_2$, where

$$\begin{array}{lcl} \mathcal{I}'_0 & = & \{P(\{\texttt{1}\},\{\texttt{2}\})\} \\ \mathcal{I}'_1 & = & \{P(\{\texttt{1}\},\{\texttt{2}\}), P(\{\texttt{2}\},\{\mathrm{v}_1^{\perp}\})\} \\ \mathcal{I}'_2 & = & \{P(\{\texttt{1}\},\{\texttt{2},\mathrm{v}_1^{\perp}\}), P(\{\texttt{2},\mathrm{v}_1^{\perp}\},\{\texttt{2},\mathrm{v}_1^{\perp}\})\}. \end{array}$$

Consequently, $chase(\mathcal{K}, \sigma') = \mathcal{I}'_2$. Thus, further investigation is needed when both finite and infinite chase sequences exist for a KB. We leave this as a topic for future work.

6 Related Work

From the extensive literature on entity resolution, and due to space limitations, we comment briefly on only a small subset of earlier work that is related to ours.

Swoosh (Benjelloun et al. 2009) is a generic approach to entity resolution in which the functions used to compare and merge records are "black boxes". In our framework, the match function is determined by entity-egds and value-egds, whereas the merge function is implemented via the union operation - an important special case of merge functions in the Swoosh approach. In this sense, our framework is less general than Swoosh. In another sense, our framework is more general as it supports tgds, differentiates between entities and values, and incorporates query answering.

As in the works on matching dependencies (MDs) (Fan 2008; Bertossi, Kolahi, and Lakshmanan 2013; Bahmani et al. 2012), we consider variables in the head of egds to be matched and merged rather than to be made equal. In (Bertossi, Kolahi, and Lakshmanan 2013), generic functions obeying some natural properties are used to compute the result of a match, while we use the union of the sets of values or sets of entities involved in the match. In the MDs framework, a version of the chase procedure is used to resolve dependency violations. This chase acts locally on a pair of tuples at each step, whereas our chase matches entities in a global fashion and values in a local fashion. Furthermore, we support tgds, which are not in the framework of MDs.

There is a body of work on declarative entity resolution and its variants that is related to our framework; relevant references include the Dedupalog framework (Arasu, Ré, and Suciu 2009), the declarative framework for entity linking in (Burdick et al. 2016; Burdick et al. 2019), and the more recent LACE framework (Bienvenu, Cima, and Gutiérrez-Basulto 2022). In both Dedupalog and LACE, there is a distinction between hard and soft entity resolution rules. LACE supports global merges and creates equivalence classes as we do, but it does not support local merges. A more recent extension, called LACE+ (Bienvenu et al. 2023) (which was not yet published at the time of our work), combines both global merges and local merges, as in our framework. Some important differences are that LACE+ combines entity resolution rules with denial constraints, while our framework combines entity resolution rules with tgds. Furthermore, they consider the complexity of various problems such as the existence of solutions and deciding whether a merge is certain or not, while we focus on the chase and on the query answering problem. The declarative framework for entity linking in (Burdick et al. 2016; Burdick et al. 2019) uses key dependencies and disjunctive constraints with "weighted" semantics that measure the strength of the links. The key dependencies are interpreted as hard rules that the solutions must satisfy in the standard sense. Consequently, two conflicting links cannot co-exist in the same solution, hence that approach uses repairs to define the notion of the certain links. In contrast, in our framework we carry along via the chase all the alternatives as either sets of values or equivalence classes of entities. Another feature of our framework is the focus on the certain answers of queries. Finally, the aforementioned declarative framework for entity linking uses more general link relations without rules for an equivalence relation of entities; thus, it is less focused on entity resolution.

7 Conclusions and Future Work

The main contribution of this paper is the development of a new declarative a framework that combines entity resolution and query answering in KBs. This is largely a conceptual contribution because the development of the new declarative framework entailed rethinking from first principles the definitions of such central notions as assignment, homomorphism, satisfaction of tgds and egds in a model, and universal solution. At the technical level, we designed a chase procedure that never fails, and showed that, when it terminates, it produces a universal solution that, in turn, can be used in query answering.

As regards future directions, perhaps the most pressing issue is to identify a "good" notion of the result of the chase when the chase procedure does not terminate. This may lead to extending the framework presented here to settings where all universal solutions are infinite. While infinite universal solutions cannot be materialized, such solutions have been used to obtain rewritability results (Calvanese et al. 2007), occasionally combined with partial materialization of the result of an infinite chase (Lutz, Toman, and Wolter 2009). In parallel, it is important to identify structural conditions on the tgds and the egds of the TBox that guarantee termination of the chase procedure in polynomial time and, thus, yield tractable conjunctive query answering. It is also worthwhile enriching our framework with other kinds of axioms, e.g., denial constraints as in (Bienvenu, Cima, and Gutiérrez-Basulto 2022), and exploring whether other variants of the chase procedure, such as the semi-oblivious (a.k.a. Skolem) chase (Marnette 2009; Calautti and Pieris 2021) or the core chase (Deutsch, Nash, and Remmel 2008), can be suitably adapted to our framework so that their desirable properties carry over.

Finally, we note that there are several different areas, including data exchange (Fagin et al. 2005), data integration (Lenzerini 2002; Doan, Halevy, and Ives 2012), ontology-mediated query answering (Bienvenu and Ortiz 2015), and ontology-based data access (Calvanese et al. 2018), in which tgds and egds play a crucial role. We believe that the framework presented here makes it possible to infuse entity resolution into these areas in a principled way.

Acknowledgments

Kolaitis' research was partially supported by NSF Award No. 1814152. Lembo and Scafoglieri were supported by EU ICT-48 2020 project TAILOR (No. 952215), EU ERA-NET Cofund ICT-AGRI-FOOD project ADCATER (No. 40705), and PNRR MUR project PE0000013-FAIR.

References

Arasu, A.; Ré, C.; and Suciu, D. 2009. Large-scale deduplication with constraints using dedupalog. In *Proc. of the 25th IEEE Int. Conf. on Data Engineering (ICDE)*, 952–963.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition.

Baget, J.; Leclère, M.; Mugnier, M.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artificial Intelligence* 175(9-10):1620–1654.

Bahmani, Z.; Bertossi, L. E.; Kolahi, S.; and Lakshmanan, L. V. S. 2012. Declarative entity resolution via matching dependencies and answer set programs. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *Proc. of the 13th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 380–390.

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

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

Bertossi, L. E.; Kolahi, S.; and Lakshmanan, L. V. S. 2013. Data cleaning and query answering with matching dependencies and matching functions. *Theoretical Computer Science* 52(3):441–482.

Bienvenu, M., and Ortiz, M. 2015. Ontology-mediated query answering with data-tractable description logics. In Faber, W., and Paschke, A., eds., *Reasoning Web. Semantic Technologies for Intelligent Data Access – 11th Int. Summer School Tutorial Lectures (RW)*, volume 9203, 218–307.

Bienvenu, M.; Cima, G.; Gutiérrez-Basulto, V.; and Ibáñez-García, Y. 2023. Combining global and local merges in logic-based entity resolution. In *Proc. of the 20th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*.

Bienvenu, M.; Cima, G.; and Gutiérrez-Basulto, V. 2022. LACE: A logical approach to collective entity resolution. In *Proc. of the 41st ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*, 379–391.

Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2016. A declarative framework for linking entities. *ACM Trans. on Database Systems* 41(3):17:1–17:38.

Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2019. Expressive power of entity-linking frameworks. *J. of Computer and System Sciences* 100:44–69.

Calautti, M., and Pieris, A. 2021. Semi-oblivious chase termination: The sticky case. *Theoretical Computer Science* 65(1):84–121.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. of Artificial Intelligence Research* 48:115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *J. of Web Semantics* 14:57–83.

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

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2018. Ontology-based data access and integration. In Liu, L., and Özsu, M. T., eds., *Encyclopedia of Database Systems, Second Edition*. Springer.

Cuenca Grau, B.; Horrocks, I.; Krötzsch, M.; Kupke, C.; Magka, D.; Motik, B.; and Wang, Z. 2013. Acyclicity notions for existential rules and their application to query answering in ontologies. *J. of Artificial Intelligence Research* 47:741–808.

Deutsch, A.; Nash, A.; and Remmel, J. B. 2008. The chase revisited. In *Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, 149–158.

Doan, A.; Halevy, A. Y.; and Ives, Z. G. 2012. *Principles of Data Integration*. Morgan Kaufmann.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theoretical Computer Science* 336(1):89–124.

Fan, W. 2008. Dependencies revisited for improving data quality. In *Proc. of the 27th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, 159–170.

Grahne, G., and Onet, A. 2018. Anatomy of the chase. *Fundamenta Informaticae* 157(3):221–270.

Johnson, D. S., and Klug, A. C. 1984. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences* 28(1):167–189.

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

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, 233–246.

Lutz, C.; Toman, D.; and Wolter, F. 2009. 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)*, 2070–2075.

Marnette, B. 2009. Generalized schema-mappings: from termination to tractability. In *Proc. of the 28th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, 13–22.

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