Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

On Referring Expressions in Query Answering over First Order Knowledge Bases

Alexander Borgida
Department of Computer Science
Rutgers University, New Brunswick, USA
borgida@cs.rutgers.edu

David Toman
Cheriton School of Computer Science
University of Waterloo, Canada
david@uwaterloo.ca

Grant Weddell
Cheriton School of Computer Science
University of Waterloo, Canada
gweddell@uwaterloo.ca

Abstract

A referring expression in linguistics is any noun phrase identifying an object in a way that will be useful to interlocutors. In the context of a query over a first order knowledge base $\mathcal{K}$, *constant symbols* occurring in $\mathcal{K}$ are the artifacts usually used as referring expressions in certain answers to the query. In this paper, we begin to explore how this can be usefully extended by allowing a class of more general formulas, called *singular referring expression*s, to replace constants in this role. In particular, we lay a foundation for admitting singular referring expressions in certain answer computation for queries over $\mathcal{K}$. An integral part of this foundation are characterization theorems for identification properties of singular referring expressions for queries annotated with a domain specific language for referring concept types. Finally, we apply this framework in the context of tractable description logic dialects, showing how identification properties can be determined at compile-time for conjunctive queries, and how off-the-shelf conjunctive query evaluation for these dialects can be used in query evaluations, preserving, in all cases, underlying tractability.

1. Introduction and Motivation

Query answering in logic-based approaches to data and knowledge bases has traditionally been viewed as finding constant names, appearing in the knowledge-base, which can be substituted for the variables of the query. More formally, a query $q(x_1, \ldots, x_n)$ is viewed as a formula with free variables $x_1, \ldots, x_n$ and, if the knowledge-base $\mathcal{K}$ contains individual constant names IN, query answering consists of computing the set $\{\, (a_1, \ldots, a_n) \mid a_i \in \mathsf{IN}, \mathcal{K} \models q(a_1/x_1, \ldots, a_k/x_k) \,\}$. We believe that in a number of circumstances this is less than ideal.

(1) In object-based KBMSs (including Object-Relational, XML and Object-Oriented DBMSs, as well as DLs with UNA), all known individual objects must have unique (internal) distinguishing identifiers. However, these identifiers are often insufficient to allow users to figure out what real-world object they refer to, especially for large KBs. For example, system generated `ref` expressions in object-oriented databases (Silberschatz, Korth, and Sudarshan 2005) and blank node identifiers in RDF are semantically opaque to end-users. A specific example of this are identifiers that individual authors or the system must invent in community-developed ontologies such as Freebase (Bollacker et al. 2008). There, for example, the `id` of the "Synchronicity" album by the Police is `"/guid/9202a8c04000641f8000000002f9e349"` (as of April, 2015.)

(2) In Relational DBMSs, the above problem is supposedly avoided by using "external keys": tuples of attributes whose values (strings, integers, ...) uniquely identify rows of tables. Problem (1) above will then arise in OBDA access to legacy relational systems, since the ontology will surely be object-based.

Even for standard databases, universally unique keys are often impossible to find (e.g., newly arrived foreign students do not have *ssn#*), though they may work for subsets of individuals, such as those returned by queries. For this reason, many tables use as keys columns labeled with suffix `_ID` (e.g., `emp_ID`), supported by "auto-increment" feature available in all modern RDBMS. Such identifiers will further obscure the meaning of answers, especially when OBDA is used to merge databases.

(3) Additional problems for finding identifying attributes for classes of objects arise in conceptual modeling. For example, consider all cases where Extended Entity-Relationship modeling creates a new heterogeneous entity set by "generalization" (Elmasri and Navathe 2000). For example, we want to generalize *Person* (with key *ssn#*) and *Company* (with key *tickerSymbol*) to *LegalEntity*, which can own things. In EER modeling, such a situation forces the introduction of a new, artificial attribute as a key, with the attendant problems. Yet when we retrieve a set of legal entities, we can reference them in different, more natural ways, depending on which subclass they belong to.

(4) The next example illustrates a subtler version of the above: consider a situation where *Publication* is a class, with subclass *EditedCollection* identified by *isbn#*, while *Journal* is in turn a subclass of *EditedCollection*, identified by *title* and *publisher*. When a query returns instances of *Publication*, there would be a natural preference for journals to be described using more meaningful $(title, publisher)$ pairs over *isbn#* for edited collections.

(5) Many kinds of KBMSs, including those based on DLs and FOL, allow one to describe situations where objects

can be inferred to exist, without having an explicit (internal) identifier. For example, if Fred is a person, then he has a mother and possibly a spouse, who are also persons. Normally, such objects cannot be returned in the list of answers unless they are named constants. This is all the more unpleasant if we can capture information about this unknown person, such as the fact that Fred's mother or spouse has a particular age, and a query searches for people of some age. Yet it is common in human communication to identify objects by their relationship to other known objects. For example, "Fred's mother" is a perfectly reasonable *referring description* of someone who is a person.

The standard response to some of the above problems would be to have the user modify the query by finding the appropriate values for identifying attributes (external keys). For example, instead of the query $\texttt{q1(x)} : -\texttt{Journal(x)}$, the programmer would be expected to write
$\texttt{q2(t,p)} : -\texttt{Journal(x)}, \texttt{hasTitle(x,t)}, \texttt{hasPublisher(x,p)}$.

This approach has several problems: (i) In the enumeration of answers to q2, the relationship between the original object of interest, x, and its descriptors, t and p, is lost; something akin to *"objects x with* title = t *and* publisher = p*"* would be more desirable. (ii) The above reformulation cannot be done using regular conjunctive queries in the case of item 4 above, because the answer for publications that are not journals should be identified by *isbn#*, for which the query is $\texttt{q5(ib)} : -\texttt{EditedCollection(x)}, \neg\texttt{Journal(x)}, \texttt{isbn(x,ib)}$ which is not a conjunctive query, since it includes a negation. (iii) From the point of view of software engineering, the task of choosing these identifying references is *mentally distinct* from the task of selecting the objects of interest to begin with. Both SQL's **select** clause, and XQuery's **return** clause are examples of separating these two aspects in existing query languages.

This paper is then dedicated to the task of proposing a *first solution* to (some of) the issues raised by providing "*singular* referring expressions" in the place of individuals returned by queries, in the context of FOL KBs, and then specialize this to conjunctive queries for some "lightweight" dialects of *description logic* (DL). (In general, referring expressions will correspond to formulas with a single free variable; in DLs these correspond naturally to concepts, if one also admits nominals.)

Our plan and contributions are as follows: We will start by proposing a language for referring expressions and types. This language will generalize the usual case of presenting answers to queries as individual names to situations that: (i) allow object identification by key (paths), possibly within the limited context of some concept instances; this will also support "descriptive answers". (ii) deal with heterogeneous answer sets, such as *LegalEntity*; and (iii) allow preferential choice of referring expressions, as for *EditedCollection*. Ultimately, we will use this language to define answers for conjunctive queries over DL knowledge bases. In our case, the query *head* will annotate each variable x returned with a *referring expression type*; as an answer, this will be instantiated to a *referring expression* in the form of a formula with a single free variable x; such formulas will eventually bottom out to individual constants.

Because we wish to generalize the usual case of constant names in answers, we desire to know when referring expression types always lead to referring expression formulas in answers that are *singular expressions*, i.e., ones which uniquely identify **one** individual.[1] Unfortunately, without knowing anything else, it is impossible, for example, to tell whether an expression such as "object with p-value 3" will be singular or not: if p is a key, then yes, but not otherwise. Therefore, we need to use information from the ontology to verify the singularity of a referring concept type. This can be extended by also examining the body of the query (and hence learning more about what kinds of values answer variables may take).

The paper is organized as follows. In Section 2, we introduce our referring expression type language within the general framework of FOL, and consider how compile-time analysis using a given ontology and query body can ensure the above-mentioned singularity property. The section then considers a stronger notion of singularity that ensures syntactically distinct singular expressions for a query variable must always denote distinct individuals. Note that this provides a basis for *counting*, that is, for aggregate queries that must compute at least a minimum number of distinct answers. Section 2 ends with a formulation of query evaluation in this general framework in terms of oracles capable of "traditional" query answering with constant names. In Section 3, we apply this general framework to a pair of tractable DL dialects, which have efficient algorithms for reasoning about logical consequence and for conjunctive query evaluation. The first is a DL-Lite dialect with path-based identification constraints (Calvanese et al. 2008a), and the second a feature-based DL dialect called $\mathcal{CFD}_{nc}^{\forall}$ that includes a capability for capturing a variety of functional dependencies based on feature paths (Toman and Weddell 2014a). In both cases, we show how tractability is preserved in our more general setting: of compile time type checking for our referring expression type language, and of run time conjunctive query evaluation. Our summary comments and conclusions then follow in Section 4.

2. Referring Expressions and Certain Answers

We assume a knowledge base corresponds to a first-order theory over a common universal signature, and that this signature consists, in turn, of constant symbols and of unary and binary predicate symbols. Also, we write D to denote a distinguished subset of the constants and assume every knowledge base includes the sentences $a \neq b$ for every distinct a and b occurring in D. Intuitively, D elements will usually correspond to elements in so-called *concrete domains*,

[1] Researchers interested in so-called *co-operative query answering* have considered *returning predicates describing sets of individuals* (e.g., (Bergamaschi, Sartori, and Vincini 1995; Borgida 1995; Imielinski 1987; Motro 1994)), where an answer to the query "Who can take the Data Structures course?" might include, "Anyone who has passed the Intro to Computer Science course with at least a C grade". Please note that we are not considering that problem in this paper.

such as integers or strings, and other objects with the unique name assumption (UNA) which we are willing to see in answers. The standard Tarskian semantics is also assumed, and, for sentences, is defined with respect to interpretations $\mathcal{I} = (\triangle, (\cdot)^{\mathcal{I}})$.

In our general setting, a query corresponds to an arbitrary well-formed formula ψ over the common signature; we write $\psi\{x_1, \ldots, x_k\}$ to also indicate the free variables.

In current practice, the space of possible answers to ψ over a given knowledge base $\mathcal{K}$ will correspond to a substitution $(x_1/a_1, \ldots, x_k/a_k)$ that maps each free variable of ψ to a constant symbol. Also recall that $(x_1/a_1, \ldots, x_k/a_k)$ qualifies as a *certain answer* when $\mathcal{K} \models \psi(x_1/a_1, \ldots, x_k/a_k)$. Our objective, however, is to study how more general well-formed formulas ϕ over the common signature can replace the individual constants a_i in substitutions thus serving the role of *singular referring expressions*, still identifying individuals. We accommodate this by the following generalization of what constitutes a space of possible answers, of certain answers, and of when certain answers are also *singular*:

Definition 1 (Certain and Singular Answers)
A *referring expression* is a formula ϕ with a single free variable over the common signature. The space of possible R-answers for a given query $\psi\{x_1, \ldots, x_k\}$ is given by an *R-substitution* θ of the form

$$\{x_1 \mapsto \phi_1\{x_1\}, \ldots, x_k \mapsto \phi_k\{x_k\}\}. \quad (1)$$

We write $\mathcal{C}_\theta$ as shorthand for $\bigwedge_{0<i\leq k} \phi_i\{x_i\}$ (where the underlying query will be clear from context).

Let $\mathcal{K}$ be a consistent knowledge base. Then θ is a *certain R-answer* with respect to $\mathcal{K}$ iff it satisfies the following two conditions:

$$\mathcal{K} \models \exists x_1, \ldots, x_k : (\mathcal{C}_\theta \wedge \psi) \quad (2)$$

and

$$\mathcal{K} \models \forall x_1, \ldots, x_k : (\mathcal{C}_\theta \rightarrow \psi). \quad (3)$$

θ is *singular* iff it is a certain R-answer[2] and satisfies a third condition for each $0 < j \leq k$:

$$\mathcal{K} \models \forall x_1, \ldots, x_{j-1}, y_1, y_2, x_{j+1}, \ldots, x_k : \\ ((\mathcal{C}_\theta \wedge \psi)(x_j/y_1) \wedge (\mathcal{C}_\theta \wedge \psi)(x_j/y_2)) \rightarrow (y_1 = y_2).$$

□

For example, in current practice, each referring expression ϕ_i in (1) has the form "$x_i = a_i$" for some constant symbol a_i. It can be paraphrased as "the object equal to a_i". A more complex referring expression might be $hasAge(x_i, 25)$ or $\exists y : hasFather(x_i, y) \wedge hasLastName(y, \text{"Castor"})$, which would be paraphrased as "the object whose age is 25" and "the object for whom the last name of the father is Castor", respectively. Of course there may be several such objects, so "the" should be "an".

[2]Henceforth we will use "certain answer" for "certain R-answer" in the rest of the paper.

A possible answer/R-substitution is really just a syntactic notion. Conditions (2) and (3) tie it to the meaning of answering query ψ in knowledge base $\mathcal{K}$, in particular, (3) disallows those R-substitutions which do not satisfy the query, while (2) eliminates the possibility that (3) is trivially true because the antecedent is false, in particular, the possibility that $\exists x_i : \phi\{x_i\}$ is false for some i.

Note that condition (2) alone will ensure that, as expected, $\mathcal{K} \models \psi(x_1/a_1, \ldots, x_k/a_k)$ in the special cases in which each of the ϕ_i have the form "$x_i = a_i$".

Now consider a query $\psi\{x\}$ and R-substitution $\{x \mapsto Person(x)\}$. If $\mathcal{K}$ consists of the axiom $\forall x : (Person(x) \rightarrow \psi\{x\})$, then all persons are answers to the query in the context of $\mathcal{K}$, while (2), i.e., the sentence $\mathcal{K} \models \exists x : Person(x) \wedge \psi$, guarantees that there is at least one answer of this form. Observe in this case, however, that $Person(x)$ is not likely to satisfy the final singularity condition on certain R-answers, and is therefore not likely to uniquely identify an individual, a "key" issue that is our main focus.

We address this issue by developing a framework in which the variety of referring expressions occurring in certain answers may be controlled by attaching so-called *referring expression types* to the free variables of a query. This is necessary in order to obtain feasible cases for the problem of computing singular R-answers, and will also allow us to characterize compile-time identification properties of such typed queries with respect to knowledge bases that consist of an *ontology* or a *mediated database schema*, that is, knowledge bases $\mathcal{K}$ in which ground data is left out. For the remainder of the paper, we write $\mathcal{K}$ to refer to such cases, and write $\mathcal{K} \cup \mathcal{K}'$ when referring to a full knowledge base consisting of an ontological component $\mathcal{K}$ and a data component $\mathcal{K}'$.

Definition 2 (Referring Expression Types)
A *referring expression type* Rt is a recursive *pattern* that starts from individual names (denoted by "?" in the grammar below) and uses paths, conjunctions and conditionals to build more complex patterns for allowed referring expressions. For example, the pattern $Person \rightarrow (name.last = \{?\} \wedge phone\# = \{?\}$ indicates that an object x in class $Person$ may be identified by a pair of constants s and b such that $\exists y.(name(x, y) \wedge last(y, s)) \wedge phone\#(x, p)$. In the grammar below A_i and R_j denote unary and binary predicate symbols, respectively:

$$Pd ::= id \mid R_1.\cdots.R_m \quad \texttt{/* paths */}$$
$$Cs ::= \{?\} \mid \langle ? \rangle \quad \texttt{/* individual matcher */}$$
$$T ::= \{A_1, \ldots, A_n\} \quad \texttt{/* conjoined concepts */}$$
$$Rt ::= Pd = Cs \mid Rt_1 \wedge Rt_2 \mid T \rightarrow Rt \mid Rt_1; Rt_2$$

A referring expression type Rt is *homogeneous* if it is free of any occurrence of the constructor ";", which expresses preference in descriptions.

We define the semantics of Rt with respect to an ontological knowledge base $\mathcal{K}$. In particular, we write $\text{RE}(Rt, \mathcal{K})$ to associate a set of referring expressions with the free variable x to each Rt, in the context of $\mathcal{K}$. In defining $\text{RE}(Rt, \mathcal{K})$

and for the remainder of the paper, we assume the following notation for various well-formed formulas:

- $T(x)$ to denote $\bigwedge_{A \in T} A(x)$;
- $Pd(x, y)$ to denote $x = y$ when $Pd =$ "id", and

  $\exists z_1, \ldots, z_n : (R_1(x, z_1) \wedge (\bigwedge_{1<i\leq n} R_i(z_{i-1}, z_i)) \wedge z_n = y)$

  when $Pd =$ "$R_1.\cdots.R_m$"; and
- for homogeneous Rt, $Rt(x, y)$ to denote $Pd(x, y)$ when $Rt =$ "$Pd = Cs$", $Rt_1(x, y) \wedge Rt_2(x, y)$ when $Rt =$ "$Rt_1 \wedge Rt_2$", and $T(x) \wedge Rt_1(x, y)$ when $Rt =$ "$T \rightarrow Rt_1$".

In the following, we assume $[Cs]$ denotes all constant symbols when $Cs =$ "$\{?\}$", and all constants in D when $Cs =$ "$\langle ? \rangle$". We also use S_i as shorthand for $\text{RE}(Rt_i, \mathcal{K})$.

$\text{RE}(Rt, \mathcal{K})$ is inductively defined as follows:

1. $\text{RE}(Pd = Cs, \mathcal{K}) =$
   $\{\exists y : (Pd(x, y) \wedge y = a) \mid a \in [Cs]\}$[3]
2. $\text{RE}(Rt_1 \wedge Rt_2, \mathcal{K}) = \{\phi_1 \wedge \phi_2 \mid \phi_1 \in S_1 \wedge \phi_2 \in S_2\}$;
3. $\text{RE}(T \rightarrow Rt_1, \mathcal{K}) = \{T(x) \wedge \phi \mid \phi \in S_1\}$; and
4. $\text{RE}(Rt_1; Rt_2, \mathcal{K}) =$
   $S_1 \cup \{\phi_2 \in S_2 \mid \neg\exists \phi_1 \in S_1 \text{ s.t. } \mathcal{K} \models \forall x : (\phi_1 \equiv \phi_2)\}$.

□

Definition 3 (Typed Queries)
Let $\psi\{x_1, \ldots, x_k\}$ be a query. A *head* for ψ, written $\text{HD}(\psi)$, associates a referring expression type with each free variable of ψ:

$$\{x_1 : Rt_1, \ldots, x_k : Rt_k\}.$$

We say that a query is *typed* if it has a head. A certain answer θ for a typed query with respect to a consistent knowledge base $\mathcal{K}$ must also satisfy the condition that $\phi\{x_i/x\} \in \text{RE}(Rt_i, \mathcal{K})$ whenever $(x_i \mapsto \phi\{x_i\}) \in \theta$. □

Before proceeding any further, we illustrate the use of referring concepts/types to resolve the issues raised in the introduction. A conjunctive query ψ with a head $\text{HD}(\psi)$ will be written in an SQL-like style "**select** $\text{HD}(\psi)$ **where** ψ":

Q.1 (expressing the current case) *"Places in which journals are published"*

select $x_2 : \{?\}$
where $\exists x_1 : Journal(x_1) \wedge publishedIn(x_1, x_2)$

Q.2 (reference via single key) *"The ssn# of any person with phone 12345567"*

select x : *ssn#* $= \{?\}$
where $Person(x) \wedge phone\#(x, 1234567)$

Q.3 (multiple attribute key) *"The title and publisher of any journals"*

select $x : title = \{?\} \wedge publishedBy = \{?\}$
where $Journal(x)$

[3]When Pd is id, we abbreviate $x = y \wedge y = a$ as a in answers.

Q.4 (choice of identification in heterogeneous set) *"Any legal entity"*

select $x : Person \rightarrow$ *ssn#* $= \{?\}$;
$Company \rightarrow tickerSymbol = \{?\}$
where $LegalEntity(x)$

For this query, a certain answer might be

$$\{x \mapsto Person(x) \wedge \textit{ssn\#}(x, 7654)\}$$

while another might be

$$\{x \mapsto Company(x) \wedge tickerSymbol(x, \text{"IBM"})\}.$$

Q.5 (preferred identification) *"Any publication, identified by its most specific identifier, when available."*

select $x : Journal \rightarrow (title = \{?\} \wedge publisher = \{?\})$;
$EditedCollection \rightarrow$ *isbn#* $= \{?\}$; $\{?\}$
where $Publication(x)$

Q.6 (multiple forms of reference, including intensional) *"Any person"*

select $x : \{?\}$; *spouseOf* $= \{?\}$;
motherOf $= \{?\}$; *fatherOf* $= \{?\}$
where *Person*(x)

In this case, a certain answer such as

$$\{x \mapsto \textit{fatherOf} = \textit{Fred}\}$$

is a particularly good illustration of answers to queries that describe individuals whose identity is not known but can be inferred to exist and have the property of being a person, that is, of answers that would not be returned in the standard setting.

These examples also illustrate the need to determine at compile-time that certain answers for a query should always uniquely identify individuals. In particular, one should be assured, regardless of the data component of a knowledge base, that people are uniquely identified by *ssn#* values in query (2), that journals are uniquely identified by a combination of $title$ values and $publishedBy$ values in query (3), and so on. We formally characterize this requirement in terms of the ontological component $\mathcal{K}$ of a knowledge base by linking the notion of a certain answer with the stronger notion of a singular answer, as we have defined them:

Definition 4 (Weak Identification)
Let $\mathcal{K}$ be a consistent knowledge base and ψ a typed query. Then $\text{HD}(\psi)$ *is weakly identifying for* ψ *with respect to* $\mathcal{K}$ iff, for all $\mathcal{K}'$ consistent with $\mathcal{K}$, every certain answer θ of ψ with respect to $\mathcal{K} \cup \mathcal{K}'$ is also singular. □

Note that this formulation of weak identification is not algorithmic. We now address this issue, starting with the introduction of a normal form for referring expression types. The normal form will lead to a formulation of weak identification as an inference problem in FOL. (Later on in Section 3, we show how this problem reduces to inference problems and knowledge base consistency checks for DL dialects and conjunctive queries.)

Lemma 5 (Normal Form)
For every referring expression type Rt, there is an *equivalent normal form*

$$Rt_1; \cdots; Rt_n,$$

denoted $\textsc{Norm}(Rt)$, that consists of *tagged record types* Rt_i that are, in turn, homogeneous referring concept types of the form

$$T_i \to (Pd_{i,1} = Cs_{i,1}) \wedge \cdots \wedge (Pd_{i,m_i} = Cs_{i,m_i}).$$

Here, the ";" referring type constructor is assumed to be left associative, and *equivalent* means that, for any knowledge base $\mathcal{K}$, $\textsc{Re}(Rt, \mathcal{K})$ coincides with $\textsc{Re}(\textsc{Norm}(Rt), \mathcal{K})$.

A typed query ψ is *normalized* whenever $(x : Rt) \in \textsc{Hd}(\psi)$ implies Rt is in normal form.

Proof (sketch): By associativity of ";" and by a straightforward induction involving the application of the following equivalence preserving rewrites.

$$\begin{array}{rcl}
Pd = Cs & \rightsquigarrow & \{\,\} \to (Pd = Cs) \\
(T \to Rt_1) \wedge Rt_2 & \rightsquigarrow & T \to (Rt_1 \wedge Rt_2) \\
Rt_1 \wedge (T \to Rt_2) & \rightsquigarrow & T \to (Rt_1 \wedge Rt_2) \\
(Rt_1 \,;\, Rt_2) \wedge Rt_3 & \rightsquigarrow & (Rt_1 \wedge Rt_3) \,;\, (Rt_2 \wedge Rt_3) \\
Rt_1 \wedge (Rt_2 \,;\, Rt_3) & \rightsquigarrow & (Rt_1 \wedge Rt_2) \,;\, (Rt_1 \wedge Rt_3) \\
T \to (Rt_1 \,;\, Rt_2) & \rightsquigarrow & (T \to Rt_1) \,;\, (T \to Rt_2) \\
T_1 \to (T_2 \to Rt) & \rightsquigarrow & (T_1 \cup T_2) \to Rt
\end{array}$$

□

Thus, a normalized typed query will associate a sequence of tagged records $Rt_{i,1}; \cdots; Rt_{i,n_i}$ with each free variable x_i in a query head. In order to deal only with "homogeneous queries" in some proofs, we want to consider all possible combinations of homogeneous types, one for each variable x_i.

Definition 6 (Induced Homogeneous Heads)
Let $\psi\{x_1, \ldots, x_k\}$ be a normalized typed query, where $\textsc{Hd}(\psi) = \{x_1 : Rt_1, \ldots, x_k : Rt_k\}$, in which each Rt_i, for each $0 < i \leq k$, is given by

$$Rt_{i,1}; \cdots; Rt_{i,n_i},$$

and, in turn, each $Rt_{i,j}$, $0 < i \leq k, 0 < j \leq n_i$, by

$$T_{i,j} \to (Pd_{i,j,1} = Cs_{i,j,1} \wedge \cdots \wedge Pd_{i,j,m_{i,j}} = Cs_{i,j,m_{i,j}}).$$

For any k-tuple $\langle j_1, \ldots, j_k \rangle$ for which $0 < j_i \leq n_i$, we write $\mathsf{H}_{j_1,\ldots,j_k}$ (where ψ will be clear from context) to denote a *homogeneous head*:

$$\{x_1 : Rt_{1,j_1}, \ldots, x_k : Rt_{k,j_k}\}.$$

We write $\mathsf{H}_{j'_1,\ldots,j'_k} < \mathsf{H}_{j_1,\ldots,j_k}$ for a pair of distinct homogeneous heads whenever $j'_i \leq j_i$, for $0 < i \leq k$. □

To characterize weak identification in terms of KB reasoning, it will be useful to have the following lemma:

Lemma 7 Let $\mathcal{K}$ be a consistent knowledge base, ψ a typed query, and θ an R-substitution for ψ of the form (1) above. Then if

$$\mathcal{K} \cup \{(\exists x_1, \ldots, x_k : \mathcal{C}_\theta \wedge \psi) \wedge (\forall x_1, \ldots, x_k : \mathcal{C}_\theta \to \psi)\} \quad (4)$$

is consistent then there is a $\mathcal{K}'$ consisting of ground literals only such that θ is a certain answer of ψ with respect to $\mathcal{K} \cup \mathcal{K}'$. In addition, $\mathcal{K}'$ can be finite whenever ψ and ϕ_i are range-restricted formulas (Abiteboul, Hull, and Vianu 1995), and can consist of positive ground atoms only whenever ψ and ϕ_i are positive.

Proof (sketch): Literals in $\mathcal{K}'$ fix part of the model of (4) that makes $\mathcal{C}_\theta \wedge \psi$ true. Note that, in general, we may need a constant for every element of the domain of such a model. However, for range-restricted formulas we only need constants that correspond to Skolem constants originating from ψ and ϕ_i. For positive queries, due to their monotonicity, it is not necessary to use any negated literals. □

The lemma tells us that all answer R-substitutions that are consistent with $\mathcal{K}$ w.r.t. a typed ψ *can become* certain answers to ψ. This property is essential to proving completeness of our characterization theorem below.

The following theorem characterizes weak identification in terms of logical implication with respect to the ontological component $\mathcal{K}$ of our KB. Note that the condition must take into account the situation in which R-answers for a more preferred head completely supersede those for a less preferred head (e.g., due to closure of predicates imposed by $\mathcal{K}$). For heads that are not completely superseded by more preferred heads, singularity relies on a functional-like constraint holding in models of $\mathcal{K}$.

Theorem 8 (Characterizing Weak Identification)
Let $\mathcal{K}$ be a consistent knowledge base and $\psi\{x_1, \ldots, x_k\}$ a normalized typed query, where $\textsc{Hd}(\psi)$ is as given in Definition 6. Then $\textsc{Hd}(\psi)$ is weakly identifying for ψ with respect to $\mathcal{K}$ iff, for every homogeneous head $\mathsf{H}_{j_1,\ldots,j_k}$ and every $0 < p \leq k$, it holds that

$$\begin{array}{l}
\mathcal{K} \models \forall y_1, \ldots, y_k, z_1, \ldots z_k : (\\
\quad \psi(x_1/y_1, \ldots, x_k/y_k) \wedge \psi(x_1/z_1, \ldots, x_k/z_k) \\
\quad\quad \wedge \neg \bigvee_{\mathsf{H}_{j'_1,\ldots,j'_k} < \mathsf{H}_{j_1,\ldots,j_k}} \mathcal{C}(\mathsf{H}_{j'_1,\ldots,j'_k}) \\
\quad\quad \wedge \mathcal{C}(\mathsf{H}_{j_1,\ldots,j_k}) \\
\quad\quad \wedge \bigwedge_{0<q\leq k; q\neq p} (y_q = z_q)) \\
\quad\quad \to (y_p = z_p)
\end{array}$$

where $\mathcal{C}(\mathsf{H}_{j_1,\ldots,j_k})$ denotes

$$\begin{array}{l}
\bigwedge_{0<i\leq k} (T_{i,j_i}(y_i) \wedge T_{i,j_i}(z_i) \\
\quad \wedge \bigwedge_{0<l\leq m_{i,j_i}} (\exists w : Pd_{i,j_i,l}(y_i, w) \wedge Pd_{i,j_i,l}(z_i, w))).
\end{array}$$

Proof (sketch): Consider a head $H_{j_1,\ldots,j_k}$. Whenever the logical implication above does not hold we can find a model of $\mathcal{K}$ and a valuation for $y_1, \ldots, y_k$ and $z_1, \ldots z_k$ that disagree on the value of y_p and z_p. Due to Lemma 7 and the observation that none of the preferred heads apply to this valuation we can create $\mathcal{K}'$ such that in $\mathcal{K} \cup \mathcal{K}'$ there will be a certain answer whose p-th component is not singular since it refers to the distinct values assigned to y_p and z_p. Conversely, non-singularity of any component of an certain answer immediately yields a counterexample to the logical implication required to hold by the theorem. □

The notion of weak identification satisfies what we believe are the minimum requirements for a referring expression in a certain answer: that the expression will hold of exactly one individual for any interpretation of a knowledge base. Recall from our introductory comments, however, that a stronger notion of identification is also desirable, one in which it becomes possible to reason about equality between differing referring expressions for the same variable when comparing certain answers. To reiterate, some capacity to reason about equality is necessary, for example, in aggregate queries. We formally characterize this stronger notion of identification as follows:

Definition 9 (Strong Identification) Let $\mathcal{K}$ be a consistent knowledge base and $\psi\{x_1, \ldots, x_k\}$ a typed query. Then $\text{HD}(\psi)$ is *strongly identifying for ψ with respect to $\mathcal{K}$* iff it is weakly identifying, and, for all $\mathcal{K}'$ consistent with $\mathcal{K}$ and every pair of certain answers θ_1 and θ_2 to ψ with respect to $\mathcal{K} \cup \mathcal{K}'$ and every $0 < i \leq k$,

$$\theta_1(x_i) \text{ is syntactically distinct from } \theta_2(x_i) \qquad (5)$$

implies

$$\mathcal{K} \cup \mathcal{K}' \models \neg\exists x_i : (\theta_1(x_i) \wedge \theta_2(x_i))$$

(assuming $\theta(x)$ denotes ϕ for each $(x \mapsto \phi) \in \theta$). □

Note that condition (5) can be easily determined in our framework. In particular, introducing set notation for conjunctions in a straightforward manner reduces the condition to a simple syntactic check for set equality. Regardless, as with weak identification, this definition of strong identification is also not algorithmic, and there remains the problem of finding an equivalent formulation as inference problems in FOL. The following provides a sufficient condition:

Theorem 10 (Recognizing Strong Identification)
Let $\mathcal{K}$ be a consistent knowledge base and $\psi\{x_1, \ldots, x_k\}$ a normalized typed query, where $\text{HD}(\psi)$ is as given in Definition 6. Then $\text{HD}(\psi)$ is strongly identifying for ψ with respect to $\mathcal{K}$ if it is weakly identifying and satisfies the following two conditions for every $0 < i \leq k$:

1. $Cs_{i,j,k} =$ "$\langle ? \rangle$", for every $0 < j \leq n_i$ and $0 < k \leq m_{i,j}$, and
2. $\mathcal{K} \models \neg\exists x : (T_{i,j}(x) \wedge T_{i,l}(x))$, for every $0 < j, l \leq n_i$ when $j \neq l$.

Proof (sketch): Consider two certain answers θ_1 and θ_2 of ψ such that $\theta_1(x_i)$ is syntactically distinct from $\theta_2(x_i)$. Due to (2) the answers cannot originate from different (homogeneous) heads in $\text{NORM}(\text{HD}(\psi))$. Then, however, the answers must differ in one of their path components, say $Cs_{i,j,k} = a_1$ in θ_1 and $Cs_{i,j,k} = a_2$ in θ_2. Thus, $\text{HD}(\psi)$ being weakly identifying and $(a_1)^{\mathcal{I}} \neq (a_2)^{\mathcal{I}}$ due to (1) contradicts the possibility of finding a witness for x_i in $\mathcal{K} \cup \mathcal{K}' \models \exists x_i : \theta_1(x_i) \wedge \theta_2(x_i)$. □

Observe that condition (1) is a simple syntactic check, and that, for the tractable DL dialects considered later on, condition (2) translates as a check for knowledge base consistency.

Query Answering with Referring Expressions

We now present a characterization of query answering in our framework in terms of an *oracle*. In particular, we assume the oracle can decide, given some query $\psi\{x_1, \ldots, x_k\}$, knowledge base $\mathcal{K} \cup \mathcal{K}'$ and constants a_i, $0 < i \leq k$, if $\mathcal{K} \cup \mathcal{K}' \models \psi(x_1/a_1, \ldots, x_k/a_k)$, written

$$\text{ANS}((x_1/a_1, \ldots, x_k/a_k), \psi, \mathcal{K} \cup \mathcal{K}').$$

Recall that the equivalent R-substitution to the substitution $(x_1/a_1, \ldots, x_k/x_k)$ in our framework is given by

$$\{x_1 \mapsto (x_1 = a_1), \ldots, x_k \mapsto (x_k = a_k)\}.$$

This characterization, given by Theorem 13 below, relies on extracting so-called *induced queries* from ψ, and on a completion of a given knowledge base. In the former case, we essentially "lift" the notion of induced homogeneous heads given by Definition 6 above to define queries for which all referring expressions are tagged record types that have additional free variables and conditions for Pd-paths. The completion consists of additional sentences that introduce fresh constant symbols, when necessary, as a way of ensuring satisfaction of preference conditions in the referring expression types in $\text{HD}(\psi)$.

Definition 11 (Induced Homogeneous Queries)
Let $\psi\{x_1, \ldots, x_k\}$ be a normalized typed query. For every homogeneous head $\mathsf{H}_{j_1,\ldots,j_k}$ given by

$$\{x_1 : Rt_1, \ldots, x_k : Rt_k\},$$

where each Rt_i is in turn given by

$$T_i \rightarrow (Pd_{i,1} = Cs_{i,1} \wedge \cdots \wedge Pd_{i,\ell_i} = Cs_{i,\ell_i}),$$

we write $\psi(\mathsf{H}_{j_1,\ldots,j_k})$ to denote the following query with additional free variables $x_{i,j}$, for $0 < i \leq k$ and $0 < j \leq \ell_i$:

$$\psi \wedge \bigwedge_{0<i\leq k} (T_i(x_i) \wedge \bigwedge_{0<j\leq \ell_i} Pd_{i,j}(x_i, x_{i,j})).$$

□

Definition 12 (Knowledge Base Completion)
Let $\mathcal{K} \cup \mathcal{K}'$ be a consistent knowledge base, and $\psi\{x_1, \ldots, x_n\}$ a normalized typed query such that ψ is weakly identifying for ψ in $\mathcal{K}$. We define the *completion of $\mathcal{K} \cup \mathcal{K}'$ relative to ψ*, written $\text{COMPL}(\mathcal{K} \cup \mathcal{K}', \psi)$, as the knowledge base $\mathcal{K}''$ obtained by an exhaustive application of the following rule to an initially empty $\mathcal{K}''$:

if there exists a homogeneous head $\mathsf{H}_{j_1,\ldots,j_k}$ with tagged record type

$$T \rightarrow (Pd_1 = Cs_1 \wedge \cdots \wedge Pd_\ell = Cs_\ell)$$

and constants a_j occurring in $\mathcal{K} \cup \mathcal{K}'$ for which

$$(\mathcal{K} \cup \mathcal{K}' \cup \mathcal{K}'') \models \exists x : \bigwedge_{0<i\leq \ell} \exists y : Pd_i(x, y) \wedge (y = a_i)$$

but where there is no constant b occurring in $\mathcal{K} \cup \mathcal{K}' \cup \mathcal{K}''$ for which

$$(\mathcal{K} \cup \mathcal{K}' \cup \mathcal{K}'') \models \bigwedge_{0<i\leq \ell} \exists y : Pd_i(b, y) \wedge (y = a_i),$$

then add $\bigwedge_{0<i\leq \ell} \exists y : Pd_i(b, y) \wedge y = a_i$ to $\mathcal{K}''$, where constant b is fresh and does not occur in D.

Induced homogeneous queries and the completion of a knowledge base now enable us to present our final result of this section, a characterization of query answering that reduces the task to an oracle for "standard" query answering for certain answers over a knowledge base:

Theorem 13 (Query Evaluation)
Let $\mathcal{K} \cup \mathcal{K}'$ be a consistent knowledge base, and $\psi\{x_1, \ldots, x_n\}$ a normalized typed query such that ψ is weakly identifying for ψ in $\mathcal{K}$. Also assume $\mathcal{K}''$ is given by $\text{COMPL}(\mathcal{K} \cup \mathcal{K}', \psi)$. Then θ is a certain answer for ψ with respect to $\mathcal{K} \cup \mathcal{K}'$ iff there is some homogeneous head $\mathsf{H}_{j_1,\ldots,j_k}$ for ψ and (appealing to notation used in Definition 11 above) θ has the form

$$\bigcup_{0<i\leq k} \{x_i \mapsto T_i(x_i) \wedge \bigwedge_{0<j\leq \ell_i} \exists y : Pd_{i,j}(x_i, y) \wedge (y = a_{i,j})\}$$

in which the $a_{i,j}$ are constant symbols occurring in $\mathcal{K} \cup \mathcal{K}'$ and, if $Cs_{i,j} =$ "$\langle ? \rangle$", also in D, and for which there exists constants b_i, $0 < i \leq k$, for which the following two conditions hold:

1. $\text{ANS}(s_1, \psi(\mathsf{H}_{j_1,\ldots,j_k}), \mathcal{K} \cup \mathcal{K}' \cup \mathcal{K}'')$ returns true for the substitution s_1 given by

   $$(x_1/b_1, \ldots, x_k/b_k, x_{1,1}/a_{1,1}, \ldots, x_{k,\ell_k}/a_{k,\ell_k}),$$

   and,

2. there is no $\mathsf{H}_{j'_1,\ldots,j'_k} < \mathsf{H}_{j_1,\ldots,j_k}$ and substitution s_2 with respective forms

   $$\{x_1 : Rt'_1, \ldots, x_k : Rt'_k\},$$

   in which each Rt'_i is in turn given by

   $$T'_i \rightarrow (Pd'_{i,1} = Cs'_{i,1} \wedge \cdots \wedge Pd'_{i,\ell_i} = Cs'_{i,\ell_i}),$$

   and

   $$(x_1/b_1, \ldots, x_k/b_k, x_{1,1}/a'_{1,1}, \ldots, x_{k,\ell_k}/a'_{k,\ell_k}),$$

   in which the $a'_{i,j}$ are constant symbols occurring in $\mathcal{K} \cup \mathcal{K}'$, for which $\text{ANS}(s_2, \psi(\mathsf{H}_{j'_1,\ldots,j'_k}), \mathcal{K} \cup \mathcal{K}' \cup \mathcal{K}'')$ returns true.

Proof (sketch): Whenever conditions (1) and (2) hold the constants $b_1, \ldots, b_k$ denote a tuple of objects that are referred to by θ. Conversely, whenever θ is a certain answer, due to the completion of $\mathcal{K}$ there must be (possibly new) constant symbols $b_1, \ldots, b_k$ such that (1) holds. Moreover, for θ to be a certain answer, there cannot be another head that, for the same $b_1, \ldots, b_k$ also produces an R-answer and dominates the head corresponding to θ. Hence (2) holds. □

3. Description Logics

The Description Logics we will consider are subsets of FOL where the unary and binary predicates, called concepts and roles, are constructed from atomic names using a variety concept and role constructors. The general goal is to find languages (sets of constructors) that lead to good computational properties for problems like consistency checking or query answering. The syntax of DLs eliminates explicit FOL variables, so we use translation functions $\tau^x(C)$ and $\tau^{x,y}(R)$ to express the semantics of various constructors by reconstructing the corresponding FOL formulas. These bottom out at atomic concepts/roles like $Person$ and $hasSpouse$, for which $\tau^x(Person) = Person(x)$ and $\tau^{x,y}(hasSpouse) = hasSpouse(x, y)$.

DL KBs are separated into two parts. The first part is called the *terminology/ontology* (TBox), and has generic axioms such as *inclusion dependencies* of the form $E_1 \sqsubseteq E_2$. These translate to FOL axioms $\forall x.\tau^x(E_1) \rightarrow \tau^x(E_2)$ in which the E_i are concepts. The second part is called the *assertion box* (ABox), and provides bindings of concept and role variables to individual constants to describe the state of the world, much like a database.

We consider the application of the material developed in Section 2 to KB's described using two specific DLs, queried using conjunctive queries. (This is the standard problem of ontology-based data access for DLs.)

We do so by taking the FOL translation of the TBox as the ontological $\mathcal{K}$, the FOL translation of the ABox as the database, conjunctive queries with heads, and then show how the decision problems in Theorems 8, 10 and 13 can be translated back to decision problems over the DLs, for which there are algorithms of known good complexity. To continue with the spirit of OBDA, R-answers $x \mapsto \phi(x)$ will be represented as $x \mapsto C_\phi$, where C_ϕ is a DL concept, thereby eliminating the free variables in answers.

Description Logic DL-Lite$^{\mathcal{F}}_{core}(idc)$

We introduce DL-Lite$^{\mathcal{F}}_{core}(idc)$, a member of *DL-Lite* family for which positive existential query answering is in AC^0 for data complexity (under the unique name assumption). For more details and the relation to other *DL-Lite* logics, we refer the interested reader to (Calvanese et al. 2008a).

Definition 14 (DL-Lite$^{\mathcal{F}}_{core}(idc)$ Knowledge Bases)
DL-Lite$^{\mathcal{F}}_{core}(idc)$ uses *basic roles* R, which are either atomic roles P or their inverses P^-, with meaning $\tau^{x,y}(P^-) = \tau^{y,x}(P)$. *Basic concepts* B are either unary predicates A or expressions $\exists R$, where R is a basic role, and $\tau^x(\exists R) = \exists y.\tau^{x,y}(R)$.

TBoxes in DL-Lite$^{\mathcal{F}}_{core}(idc)$ have inclusion axioms involving basic concepts, where the subsumer may be negated.

In addition, of central interest to us are (i) *functionality axioms* $(\mathsf{funct}\ R)$ and (ii) path-based *identification assertions/constraints* $(\mathsf{id}\ B\ \pi_1, ..., \pi_k)$, where B is a basic concept, and each path π_i is of the form

$$\pi ::= R \mid \pi \circ \pi$$

for basic roles R.[4] The semantics of $(\mathsf{funct}\ R)$ is given by the assertion

$$\forall u, v, w : \tau^{u,v}(R) \wedge \tau^{u,w}(R) \rightarrow v = w.$$

The semantics of IdC's starts with semantics for paths, which defines composition over binary relations for $\circ$:

$$\tau^{x,y}((\pi_1 \circ \pi_2)) = \exists z.\tau^{x,z}(\pi_1) \wedge \tau^{z,y}(\pi_2).$$

[4] The full syntax of paths in (Calvanese et al. 2008b) also allows for role $B?$, which is identity over concept B, but we ignore these here for simplicity.

(Observe that π has exactly the same semantics as Pd in referring expression (types).) Finally, the semantics of identification constraints, (id B $\pi_1, ..., \pi_k$), is given by the following assertion:

$$\forall w, w' : (\tau^w(B) \wedge \tau^{w'}(B) \\ \wedge \bigwedge_{0<i\leq k}(\exists z : \tau^{w,z}(\pi_i) \wedge \tau^{w',z}(\pi_i))) \rightarrow w = w'.$$

□

We now extend the syntax DL-Lite$^{\mathcal{F}}_{core}(idc)$ in a natural manner to enable capturing referring expressions as concepts:

Definition 15 (Referring Expressions in DL-Lite$^{\mathcal{F}}_{core}(idc)$) The concept constructors for DL-Lite$^{\mathcal{F}}_{core}(idc)$ available to express referring concepts include the following:

1. atomic concepts A, and concept conjunction $\sqcap$, which allow the representation of $T = \{A_1, A_2, ...\}$ as the concept "$A1 \sqcap A_2 \sqcap ...$", and $\phi_1(x) \wedge \phi_2(x)$ as the concept "$\phi_1 \sqcap \phi_2$";
2. nominals $\{a\}$, whose meaning $\tau^x(\{a\})$ is $(x = a)$; and
3. qualified existential restrictions, $\exists\pi.B$, where π is a path, as introduced above. These capture referring expressions of the form
$$\exists y : Pd(x, y) \wedge y = a$$
as concept $\exists Pd.\{a\}$. (In the case when $Pd = id$, this abbreviates to $\{a\}$.) □

To illustrate, consider the two example certain answers to query **Q.4** in the previous section. The referring expressions in these answers would now be given in a variable-free form as DL-Lite$^{\mathcal{F}}_{core}(idc)$ concepts, as in

$$\{x \mapsto Person \sqcap \exists ssn\#.\{7654\}\}, \text{ and}$$

$$\{x \mapsto Company \sqcap \exists tickerSymbol.\{\text{"IBM"}\}\}.$$

Theorem 16 (Weak Identification)
Let $\mathcal{T}$ be a DL-Lite$^{\mathcal{F}}_{core}(idc)$ TBox and ψ a typed conjunctive query. Then $\text{HD}(\psi)$ is weakly identifying for ψ with respect to $\mathcal{T}$ if and only if, for each homogeneous head in $\text{NORM}(\text{HD}(\psi))$, consisting of

$$x_i : T_i \rightarrow (Pd_{i,1} = Cs_{i,1} \wedge \cdots \wedge Pd_{i,\ell_i} = Cs_{i,\ell_i}),$$

for $0 < i \leq k$, $\mathcal{K} = (\mathcal{T}, \mathcal{A}_p)$ is inconsistent for each ABox A_p $(0 < p \leq k)$, defined as

$$\mathcal{A}_p = \{\psi(x_1/a_1, \ldots, x_k/a_k), \psi(x_1/b_1, \ldots, x_k/b_k)\} \\ \cup \{T_i(a_i), T_i(b_i) \mid 0 < i \leq k\} \\ \cup \{Pd_{i,j}(a_i, c_{i,j}), Pd_{i,j}(b_i, c_{i,j}) \mid 0 < i \leq k, 0 < j \leq \ell_i\} \\ \cup \{A(a_p), \neg A(b_p)\}$$

where $\psi(x_1/d_1, \ldots, x_k/d_k)$ is a set of ABox assertions corresponding to the atoms in the body of conjunctive query ψ in which answer variables x_i were replaced by constants d_i and quantified variables by fresh distinct constants, A is a fresh primitive concept and a_i, b_i and $c_{i,j}$ distinct constants except for a_i and b_i being identical for $i \neq p$.

Proof (sketch): A model for $\mathcal{K}$ defined above provides a counterexample to singularity of $\text{HD}(\psi)$. Note that, unlike in the general FOL case, we do not need to test whether a more preferred head *captures* all answers to the head under consideration, as in DL-Lite$^{\mathcal{F}}_{core}(idc)$ concept/role closure cannot be expressed. Conversely, if a particular certain answer violated the singularity condition, we could construct an ABox $\mathcal{A}$ that would violate the condition above. Note that since such an answer exists it must conform to one of the heads in $\text{NORM}(\text{HD}(\psi))$. □

The sufficient conditions for *strong identification* required in Theorem 10 can be similarly expressed as a consistency question in DL-Lite$^{\mathcal{F}}_{core}(idc)$.

Theorem 17 (Query Answering)
Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$^{\mathcal{F}}_{core}(idc)$ knowledge base and ψ a typed conjunctive query. Then certain answers can be computed in AC^0 data complexity.

Proof (sketch): It is sufficient to observe that each of the uses of ANS in the construction in Theorem 13 is with respect to $(\mathcal{T}, \mathcal{A}')$ for $\mathcal{A}'$ a completion of $\mathcal{A}$ that depends only on $\mathcal{T}$ and ψ. Applying Theorem 13 completes the proof. □

To prove the above theorem, we reduced the problem to the general FOL case. However, in the case of description logics (and logics with the tree model property in general) it is not difficult to show that the structure of models outside of the ABox can be restricted to tree shaped structures. In such a setting, the additional constants b_i introduced in the completion of $\mathcal{A}$ can be *proxied* by already existing named objects in $\mathcal{A}$, namely those that are the last in $\mathcal{A}$ before the path Pd *exits* the ABox (as for an b, such object is unique). The full construction is beyond the limits of this paper.

Description Logic $\mathcal{CFD}^{\forall}_{nc}$

We now introduce $\mathcal{CFD}^{\forall}_{nc}$, a member of the $\mathcal{CFD}$ family of description logics that also have tractable reasoning and query answering. For more details on this dialect, we refer the reader to (Toman and Weddell 2014a).

Definition 18 ($\mathcal{CFD}^{\forall}_{nc}$ Knowledge Bases)
The syntax of $\mathcal{CFD}^{\forall}_{nc}$ starts with constants a, atomic concepts A, and features f, which are roles that must be total functions. Paths *Pf* are either id, denoting identity, or have the form $f_1.\cdots.f_k$, and have semantics aligning with that of a Pd in referring expression types and with π in DL-Lite$^{\mathcal{F}}_{core}(idc)$.
TBoxes in $\mathcal{CFD}^{\forall}_{nc}$ can have inclusion dependencies resembling any of the following forms, where B is a possibly negated atomic concept:

$$A \sqsubseteq B, \quad A \sqsubseteq \forall \mathit{Pf}.B, \quad \forall \mathit{Pf}.A \sqsubseteq B,$$
$$A \sqsubseteq (\mathit{Pf}_1 = \mathit{Pf}_2) \text{ or}$$
$$A_1 \sqsubseteq A_2 : \mathit{Pf}_1, \ldots, \mathit{Pf}_k \rightarrow \mathit{Pf}.$$

Inclusions dependencies occurring in a TBox that are given by the final two possibilities are also assumed to satisfy additional syntactic conditions: that A does not appear on the

right-hand-side of any other dependencies in the TBox in the former case, and, in the latter case, that *Pf* is a *prefix* of Pf_1 or has the form $Pf_0.f$ where Pf_0 is a prefix of Pf_1.

The semantics of the constituent concept constructors are given as follows: $\tau^x(\forall Pf.C)$ and $\tau^x(Pf_1 = Pf_2)$, respectively, by

$$\forall y : Pf(x, y) \rightarrow \tau^y(C) \text{ and}$$
$$\forall y : (Pf_1(x, y) \leftrightarrow Pf_2(x, y)),$$

and $\tau^x(A : Pf_1, \ldots, Pf_k \rightarrow Pf)$ by

$$\forall y, z_1, \ldots, z_k, w : (A(y) \wedge \textstyle\bigwedge_i (Pf_i(x, z_i) \leftrightarrow Pf_i(y, z_i))) \rightarrow (Pf(x, w) \leftrightarrow Pf(y, w))$$

A $\mathcal{CFD}_{nc}^{\forall}$ ABox contains assertions of the form $A(a)$ and so-called *path assertions* of the form $a.Pf_1 = b.Pf_2$. The semantics of path assertions, $\tau^x(a.Pf_1 = b.Pf_2)$ is given by $\exists y : (Pf_1(a_1, y) \wedge Pf_2(a_2, y))$. □

Definition 19 (Referring Expressions in $\mathcal{CFD}_{nc}^{\forall}$)
As in DL-Lite$_{core}^{\mathcal{F}}(idc)$, we add nominal concepts $\{a\}$ and concept conjunction $\sqcap$. In this case, referring expressions $\exists y : (Pd(x, y) \wedge y = a)$ are represented by concepts $\forall Pd.\{a\}$, because the universal and existential quantifiers are equivalent when all the path elements are total functions. □

To illustrate, consider again the two example certain answers to query **Q.4** in the previous section. The referring expressions in these answers would now be given in a variable-free form as $\mathcal{CFD}_{nc}^{\forall}$ concepts, as in

$$\{x \mapsto Person \sqcap \forall ssn\#.\{7654\}\}, \text{ and}$$
$$\{x \mapsto Company \sqcap \forall tickerSymbol.\{\text{“IBM”}\}\}.$$

In preparation to formulating the *weak identification* theorem for $\mathcal{CFD}_{nc}^{\forall}$, we need the following mapping of conjunctive queries to $\mathcal{CFD}_{nc}^{\forall}$ concepts, where w.l.o.g., query variables and constants are treated as functional role names:

$$\sigma(\psi) = \begin{cases} \forall\, x.C & \text{if } \psi = \text{“}C(x)\text{”}; \\ (x_1.f = x_2) & \text{if } \psi = \text{“}f(x_1, x_2)\text{”}; \\ \sigma(\psi_1) \sqcap \sigma(\psi_2) & \text{if } \phi = \text{“}\psi_1 \wedge \psi_2\text{”}; \text{ and} \\ \sigma(\psi_0) & \text{if } \psi = \text{“}\exists x_{k+1}, \ldots, \exists x_m : \psi_0\text{”}. \end{cases}$$

Theorem 20 (Weak Identification)
Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall}$ TBox and ψ a typed conjunctive query. Then $\text{HD}(\psi)$ is weakly identifying for ψ with respect to $\mathcal{T}$ if and only if

$$\mathcal{T} \cup \{A \sqsubseteq (\forall x_1.T_1 \sqcap \ldots \sqcap \forall x_k.T_k \sqcap \sigma(\psi))\} \models A \sqsubseteq A : x_i.Pf_{i,1}, \ldots, x_i.Pf_{i,m_i} \rightarrow x_i$$

for each homogeneous head H in $\text{NORM}(\text{HD}(\psi))$ consisting of

$$x_i : T_i \rightarrow (Pd_{i,1} = Cs_{i,1} \wedge \cdots \wedge Pd_{i,\ell_i} = Cs_{i,\ell_i}),$$

and for each $0 < i \leq k$, where A is a fresh primitive concept.

Proof (sketch): A counterexample to the above entailment yields a knowledge base in which a certain answer to ψ conforming to H is not singular. Conversely, a knowledge base K yielding non-singular R-answer to ψ to ψ is a counterexample to the above entailment. □

The sufficient conditions for *strong identification* required in Theorem 10 can be expressed as a logical implication question in $\mathcal{CFD}_{nc}^{\forall}$.

Theorem 21 (Query answering)
Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFD}_{nc}^{\forall}$ knowledge base and ψ a typed conjunctive query. Then computing certain answers is complete for PTIME in data complexity.

Proof (sketch): Let $\mathcal{A}'$ be an ABox obtained from $\mathcal{A}$ by explicitly naming all intermediate objects involved in path assertions $\exists x : Pf_1(a_1, x) \wedge Pf_2(a_2, x) \in \mathcal{A}$. For $\mathcal{CFD}_{nc}^{\forall}$ TBoxes and typed conjunctive queries, it is easy to see that applying KB completion to $(\mathcal{T}, \mathcal{A}')$ does not introduce any additional constants. Thus, applying Theorem 13 completes the proof for ANS given by the query answering algorithm for $\mathcal{CFD}_{nc}^{\forall}$ presented in (Toman and Weddell 2013). □

The approach can be extended to a more expressive dialect of the $\mathcal{CFD}$ family $\mathcal{CFDI}_{nc}^{\forall -}$ (Toman and Weddell 2014b), that subsumes *DL-Lite*$_{core}^{\mathcal{F}}$, with the help of Theorem 13 and a query answering algorithm for $\mathcal{CFDI}_{nc}^{\forall -}$.

4. Conclusions

The paper's contributions are as follows.

First, on the non-technical side, it recognized and motivated the utility of “singular referring expressions” (formulas with a single free variable) as query answers, which are more complex than just nominals, and it argued for the need for a new separation of concerns in query writing: qualification (what the query body searches for) vs. identification (how results are presented).

On the specification side, the paper defined formally the notion of “query answering using singular referring expressions” for certain answers to general FOL queries over FOL knowledge bases. It introduced one particular language for referring expression formulas and types (the types are necessary to limit the set of referring expression formulas to be finite) which allows us to handle all the motivating examples in Section 1. This language generalizes the notion of constant/nominal, currently used in OBDA, to handle keys (as found in both the database and DL literature), and supports (i) heterogeneous sets of values (as in the case of $LegalEntity$), (ii) preferential choice of referring expressions (as in the $EditedCollection$ example), as well as (iii) descriptive answers of individuals (as in “Fred's mother”). The latter allow new answers to be returned by queries, ones for which there were no names in the database.

The notion of “singularity of reference” is the one aspect of nominals that we wanted to carry over in this work. Therefore, in the general FOL framework, we considered how compile-time analysis, using a given ontology and query body, can ensure the singularity property. We also considered a stronger notion of singularity that ensures syntactically distinct singular expressions for a query variable must always denote distinct individuals. Finally, we reformulated query evaluation with referring expressions in terms of oracles capable of “traditional” query answering with constant names.

Finally, we applied the above general theorems to two "lightweight" DLs that can be used for OBDA and that have fast algorithms for consistency checking and conjunctive query answering. By using the oracles, we showed that query answering with referring expressions of the types introduced here can be done without any increase in data complexity.

Related work: The problem of finding referring expressions that identify an element in a given knowledge base is a basic problem in natural language generation. For example, (van Deemter et al. 2012) considers so-called incremental algorithms that address this problem (and provides an extensive review of related work). Indeed, some authors have considered the problem in the context of logic-based knowledge bases (Areces, Figueira, and Gorín 2011), including the case where the underlying logic is a description logic (Areces, Koller, and Striegnitz 2008). In finding referring expressions, the basic problem assumes that the full knowledge base is available as input. The problems we have addressed involve a referring expression type language, the compile time analysis of identification properties relative to an ontological component of a knowledge base, and of efficient query evaluation, are quite distinct from the above, and, so far as we are aware, constitute a new area of exploration. To reiterate, our framework views the issue of referring expressions as more of a programming task to be undertaken during query formulation.

For future work, an obvious problem to consider is alternative/extended referring expression (types), especially as motivated by DLs to which they can be applied.

The reader may also have observed that so far it was up to the programmer to select the referring expression(s) to consider for each variable. A form of type inference on the query variables would be useful, as the basis of a tool which would suggest to the user a (bounded) list of possible referring expressions that are guaranteed to have the singular reference property with respect to a particular TBox.

Alternatively, the ontology designer may associate with concepts referring types, and the head of the query could be constructed (semi)automatically using this information. (A similar approach, for over-loading print functions, was used in industrial applications of the CLASSIC DL.)

References

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

Areces, C.; Figueira, S.; and Gorín, D. 2011. Using logic in the generation of referring expressions. In Pogodalla, S., and Prost, J., eds., *Logical Aspects of Computational Linguistics - 6th International Conference, LACL 2011, Montpellier, France, June 29 - July 1, 2011. Proceedings*, volume 6736 of *Lecture Notes in Computer Science*, 17–32. Springer.

Areces, C.; Koller, A.; and Striegnitz, K. 2008. Referring expressions as formulas of description logic. In White, M.; Nakatsu, C.; and McDonald, D., eds., *INLG 2008 - Proceedings of the Fifth International Natural Language Generation Conference, June 12-14, 2008, Salt Fork, Ohio, USA*. The Association for Computer Linguistics.

Bergamaschi, S.; Sartori, C.; and Vincini, M. 1995. Dl techniques for intensional query answering in oodbs. In *Working Notes of the KI'95 Workshop: KRDB-95 Reasoning about Structured Objects: Knowledge Representation Meets Databases*, 3 pages.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*, 1247–1250. ACM.

Borgida, A. 1995. Description logics in data management. *Knowledge and Data Engineering, IEEE Transactions on* 7(5):671–682.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2008a. Path-Based Identification Constraints in Description Logics. In *Principles of Knowledge Representation and Reasoning*, 231–241.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2008b. Path-based identification constraints in description logics. In *KR*, 231–241.

Elmasri, R., and Navathe, S. B. 2000. *Fundamentals of Database Systems, 3rd Edition*. Addison-Wesley-Longman.

Imielinski, T. 1987. Intelligent query answering in rule based systems. *The Journal of Logic Programming* 4(3):229–257.

Motro, A. 1994. Intensional answers to database queries. *Knowledge and Data Engineering, IEEE Transactions on* 6(3):444–454.

Silberschatz, A.; Korth, H. F.; and Sudarshan, S. 2005. *Database System Concepts, 4th Edition*. McGraw-Hill Book Company.

Toman, D., and Weddell, G. E. 2013. Conjunctive Query Answering in $\mathcal{CFD}_{nc}$: A PTIME Description Logic with Functional Constraints and Disjointness. In *Australasian Conference on Artificial Intelligence*, 350–361.

Toman, D., and Weddell, G. E. 2014a. Answering Queries over $\mathcal{CFD}_{nc}^{\forall}$ Knowledge Bases. Technical Report CS-2014-14, Cheriton School of Computer Science, University of Waterloo.

Toman, D., and Weddell, G. E. 2014b. On adding inverse features to the description logic $\mathcal{CFD}_{nc}^{\forall}$. In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014.*, 587–599.

van Deemter, K.; Gatt, A.; van der Sluis, I.; and Power, R. 2012. Generation of referring expressions: Assessing the incremental algorithm. *Cognitive Science* 36(5):799–836.