Singular Referring Expressions in Conjunctive Query Answers: the case for a CFD DL Dialect

Alexander Borgida[†], David Toman[‡] and Grant Weddell[‡]

[†]Department of Computer Science Rutgers University, New Brunswick, USA borgida@cs.rutgers.edu [‡]Cheriton School of Computer Science University of Waterloo, Canada {david,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 conjunctive queries over a description logic knowledge base (DL KB), typically *constant symbols* (usually treated as rigid designators) are used as referring expressions in a certain answer to the query. In this paper, we begin to explore how this can be usefully generalized by allowing more general DL concept descriptions, called *singular referring expressions*, to replace constants in this role. In particular, we lay the foundation for singular referring expressions in conjunctive query answers over a DL KB using a member of the CFD family of DL dialects. In the process, we introduce a specific language for referring concept types, and present initial results on how conjunctive queries with referring concept types can be efficiently supported.

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 knowledgebase, 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 [10] 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 [2]. There, for example, the id of the "Synchronicity" album by the Police is "/guid/9202a8c04000641f800000002f9e349" (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.¹

We note that even in databases, universally unique keys are hard 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.

(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" [5] from others. For example, we want to generalize *Person* (whose key might be ssn#) and *Company* (whose key might be 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 the following hierarchy of concepts relating to publications:

$Journal \sqsubseteq EditedCollection \ , \ EditedCollection \sqsubseteq Publication$

And suppose edited collections are identified by isbn#, while journals are identified by *title* and *publisher*. When we retrieve a set of objects in *Publication*, we would like to describe them in different ways, depending on the subclass they belong to; but in this case, there would be an additional preference for *textittitle*, *publisher*) over isbn# for elements of *Journal*.

(5) Many kinds of KBMSs, including those based on DLs and FOL, allow us to describe situations where objects can be inferred to exist, without having an explicit (internal) identifier. For example, if Michelle is a person, then she has a mother, and if she is married then she has a spouse. Normally, such objects cannot be returned in the list of answers. This is all the more unpleasant if we can capture information about this unknown person, such as the phone number of Michelle's mother: {Michelle} $\subseteq \exists hasMother. \exists hasPhone. \{1234567\}$. Yet it is common in human communication to identify objects by their relationship to other known objects. For example, "Michelle's mother" is a perfectly reasonable *intensional description* of someone who has phone 1234567.

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 q1(x) := Journal(x), the programmer would be expected to write

q2(t, p) : -Journal(x), hasTitle(x, t), hasPublisher(x, p).

¹ OBDA systems such as MASTRO [4] and others try to deal with this issue by using function symbols over database keys to generate "object terms" that act as object identifiers. The name of the function symbols is however not semantically motivated.

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 edited collections that are not journals should be identified by isbn#, for which the query is

q5(isb): -EditedCollection(x), \neg Journal(x), hasIsbn(x, isbn) 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 conjunctive queries, in the context of DL KBs.

Our plan and contributions are as follows: We will start by proposing a language for referring concept 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; (ii) deal with heterogeneous answer sets, such as *LegalEntity*; and (iii) allow preferential choice of referring expressions, as for *EditedCollection*. We will use this language to define answers for conjunctive queries over DL KBs. More generally, the proposed approach introduces a new separation of concerns for knowledge bases (identification vs. qualification). In our case, the query head will annotate each variable returned with an *answer concept type*; this will be instantiated to an *answer concept* (a subset of our DL concepts) for each answer; such concepts will eventually bottom out to individual constants, rather than atomic concepts.

Because we wish to generalize the usual case of constant names in answers, we desire referring concept types to be singular expressions – i.e., to identify **one** individual.² 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 referring concept type. This can be extended by examining the body of the query (and hence learning more about what possible values variables may take). We will concentrate in this paper on: (i) the (compile-time) analysis of conjunctive query bodies in the context of the TBox to determine whether a referring concept *type* will return a reference to at most one individual; (ii) the reformulation of the query to

² Researchers interested in so-called *co-operative query answering* have considered *re-turning predicates/concepts describing sets of individuals* (e.g., [1, 3, 6, 8]), 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.

guarantee that the referring expression will indeed return exactly 1 value. Our technical results will show that this can be done in polynomial time for the DL CFD_{nc}^{\forall} , which allows the capture of identification constraints such as keys.

2 Preliminaries: the Description Logic $\mathcal{CFD}_{nc}^{\forall}$

The knowledge bases that we consider are based on the logic $C\mathcal{FD}_{nc}^{\forall}$, a recent member of the $C\mathcal{FD}$ family of DL dialects. All members of this family are fragments of FOL with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *attributes*. Although attributes deviate from the normal practice of using binary predicate symbols called *roles*, they make it easier to incorporate concept constructors suited to the capture of relational data sources that include various dependencies, e.g., by a straightforward reification of arbitrary *n*-ary predicates, and also make it easier to explore varieties of concepts that can serve as referring expressions.

Definition 1 (CFD_{nc}^{\forall} **Concepts**) Let F, PC and IN be disjoint sets of (names of) *attributes, primitive concepts* and *individuals,* respectively. A *path function* Pf is a word in F^{*} with the usual convention that the empty word is denoted by *id* and concatenation by ".". The set of CFD_{nc}^{\forall} concepts C is given by the following grammar, where $a \in IN$, $A \in PC$, Pf and Pf_i are path functions, k > 0 and $f_i \in F$.

 $C ::= \{a\} \mid A \mid \forall Pf.C \mid C_1 \sqcap C_2 \mid \neg A \mid Pf_1 = Pf_2 \mid A : Pf_1.Pf, Pf_2, \dots, Pf_k \to Pf_1 \mid A : Pf_1.Pf, Pf_2, \dots, Pf_k \to Pf_1.f$ (1)

Semantics is defined in the standard way with respect to an interpretation $\mathcal{I} = (\Delta, (\cdot)^{\mathcal{I}})$, where Δ is a domain of "objects" and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretation of attributes f to be total functions on Δ , primitive concepts A to be subsets of Δ , and individuals a, b to be elements of Δ . The

$$\begin{split} (\{a\})^{\mathcal{I}} &= \{(a)^{\mathcal{I}}\},\\ (\forall Pf.C)^{\mathcal{I}} &= \{x \in \Delta \mid (Pf)^{\mathcal{I}}(x) \in (C)^{\mathcal{I}}\},\\ (C_1 \sqcap C_2)^{\mathcal{I}} &= (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}},\\ (\neg A)^{\mathcal{I}} &= \Delta \setminus (A)^{\mathcal{I}},\\ (Pf_1 = Pf_2)^{\mathcal{I}} &= \{x \in \Delta \mid (Pf_1)^{\mathcal{I}}(x) = (Pf_2)^{\mathcal{I}}(x)\} \text{ and}\\ (A: Pf_1, \dots, Pf_k \to Pf)^{\mathcal{I}} &= \{x \in \Delta \mid \forall y \in (A)^{\mathcal{I}}:\\ & & \bigwedge_i (Pf_i)^{\mathcal{I}}(x) = (Pf_i)^{\mathcal{I}}(y) \to (Pf)^{\mathcal{I}}(x) = (Pf)^{\mathcal{I}}(y)\} \end{split}$$

Fig. 1. Semantics of $\mathcal{CFD}_{nc}^{\forall}$ Concepts.

interpretation function is extended to path expressions by interpreting *id* as the identity function and concatenation as function composition. The semantics of the remaining $\mathcal{CFD}_{nc}^{\forall}$ concepts are then defined in Figure 1.

Concepts having the last two forms in (1) are called a *(path) key* and a *path functional dependency* (PFD), respectively. Informally, such concepts each de-

note a set of objects, each of which, whenever agreeing with any A-object on all left-hand-side path functions, also agrees with that object on the right-hand-side path function. Thus, the axiom $EditedCollection \sqsubseteq EditedCollection : isbn# \rightarrow id$, expresses that isbn# is a key for edited collections, while $Person \sqsubseteq Person : home.phone# \rightarrow home.address$, says that if two persons have the same home phone then they have the same home address.

Definition 2 ($\mathcal{CFD}_{nc}^{\forall}$ **Knowledge Bases**) Generic knowledge/metadata and specific facts/data in a $\mathcal{CFD}_{nc}^{\forall}$ knowledge base \mathcal{K} are respectively defined by a *TBox* $\mathcal{T}_{\mathcal{K}}$ and an *ABox* $\mathcal{A}_{\mathcal{K}}$.

A TBox $\mathcal{T}_{\mathcal{K}}$ consists of a finite set of general concept inclusion axioms, which adhere to one of the following six forms, where A and A_i are primitive concepts, f is an attribute in F, B is a primitive concept or a negation of a primitive concept, and where Pf and Pf_i are path functions:

 $A \sqsubseteq B; A \sqsubseteq \forall f.B; \forall f.A \sqsubseteq B; A \sqsubseteq (Pf_1 = Pf_2); A_1 \sqsubseteq A_2 : Pf_1, \dots, Pf_k \to Pf.$ $\mathcal{T}_{\mathcal{K}}$ must also satisfy the following condition:

stratification of path function equalities: If $A \sqsubseteq (Pf_1 = Pf_2) \in \mathcal{T}_{\mathcal{K}}$ then A is a primitive concept that does not occur on the right-hand-side of any inclusion axiom in $\mathcal{T}_{\mathcal{K}}$.

An ABox $\mathcal{A}_{\mathcal{K}}$ consists of a finite set of axioms that express facts, each of which has one of the following two forms, respectively called *individual membership assertions* and *individual relationship assertions*,

(a) and
$$Pf_1(a) = Pf_2(b)$$
,

for individuals $a, b \in \mathsf{IN}$, primitive concept $A \in \mathsf{PC}$, and path functions $Pf_i \in \mathsf{F}^*$. Note that $(\{a\})^{\mathcal{I}} = \{(a)^{\mathcal{I}}\}$ and thus we can use nominal concepts as proxies for individuals. An interpretation \mathcal{I} satisfies an inclusion axiom $C_1 \sqsubseteq C_2$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$. It satisfies ABox axioms A(a) and $Pf_1(a) = Pf_2(b)$ if $(a)^{\mathcal{I}} \in (A)^{\mathcal{I}}$ and $(Pf_1)^{\mathcal{I}}((a)^{\mathcal{I}}) = (Pf_2)^{\mathcal{I}}((b)^{\mathcal{I}})$, respectively. \mathcal{I} satisfies a knowledge base \mathcal{K} if it satisfies each axiom in \mathcal{K} .

The need for the additional restriction on a TBox to avoid undecidability of TBox reasoning due to equational constraints derives in a straightforward way from the undecidability of the word problem for monoids [7, 9].

Proposition 3 (Consistency and Logical Implication in CFD_{nc}^{\forall} [11, 13]) Knowledge base consistency and logical implication for CFD_{nc}^{\forall} are complete for PTIME.

3 Conjunctive Queries and Certain Answers

A

In this section we re-evaluate the way answers to conjunctive queries are understood and presented.

Definition 4 (Conjunctive Queries) A conjunctive query (CQ) Q, with free variables $\{x_1, \ldots, x_k\}$, has the form $\exists x_{k+1}, \ldots, \exists x_m : BODY(Q)$ where BODY(Q),

the query body of Q, is a first order formula over the signature $\mathsf{C} \cup \mathsf{F}$ of the form

$$\left(\bigwedge C(x_i)\right) \land \left(\bigwedge f(x_j) = x_k\right)\right), \tag{2}$$

where each x_i, x_j and x_k occurs in $\{x_1, \ldots, x_m\}$.³

Recall that our objective in this paper is to study how $\mathcal{CFD}_{nc}^{\forall}$ concepts can help serve the role of a singular referring expression in certain answers to conjunctive queries. To review current practice, assume \mathcal{K} is a knowledge base over some DL dialect, and consider one of the purposes served by an ABox in defining the *certain answers* over \mathcal{K} to a CQ Q: the finite collection of individual names $\{a_1, \ldots, a_n\}$ in the ABox defines a space of n^k potential answers to Q: k-tuples θ_i that map query variables x_i to individual names a_{i_i} ,

$$\{x_1 \mapsto a_{i_1}, \dots, x_k \mapsto a_{i_k}\}.$$

Viewed as a substitution, recall that each θ_i is a *certain answer* to Q over \mathcal{K} exactly when $\mathcal{K} \models Q\theta_i$. Note that the occurrence of an individual a_{i_j} in a certain answer is a simple example of a *referring expression*, i.e., a syntactic artifact that identifies elements of an underlying domain.

This notion of a potential answer is easily modified to accommodate a much larger variety of referring expressions. To start, one can view a potential answer θ_i as a set of size k that maps query variables to *nominal concepts* instead of individuals, as in

$$\{x_1 \mapsto \{a_{i_1}\}, \ldots, x_k \mapsto \{a_{i_k}\}\},\$$

and then extend this idea by allowing *arbitrary* CFD_{nc}^{\forall} concepts in potential answers to queries, i.e., potentially by allowing answers to have the form

$$\{x_1 \mapsto C_1, \dots, x_k \mapsto C_k\}.$$
(3)

In this alternative setting "certain answers" are defined as follows:

Definition 5 (Referring Concepts and Certain Answers)

Let \mathcal{K} be a $\mathcal{CFD}_{nc}^{\forall}$ knowledge base, Q a conjunctive query with free variables x_1, \ldots, x_k and $C_1, \ldots, C_k \mathcal{CFD}_{nc}^{\forall}$ concept descriptions. We say that C_1, \ldots, C_k are referring concepts in a certain answer $\{x_1 \mapsto C_1, \ldots, x_k \mapsto C_k\}$ to Q if the following two conditions hold:

1.
$$\mathcal{K} \models \forall x_1, \dots, x_k.C_1(x_1) \land \dots \land C_k(x_k) \to Q$$
, and
2. $|\{o \in \Delta \mid \mathcal{I}, [x_i \mapsto o] \models C_i(x_i) \land (\exists x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k.Q)\}| = 1$
for every $\mathcal{I} \models \mathcal{K}$ and $0 < i \leq k$.

where C(x) is the first-order formula derived from the concept description C. \Box

The first condition states that C_i objects (as values of x_i s) satisfy Q and the second one that we are interested in *singular* referring expressions C_i , as generalizations of simple individual names. In the rest of the paper we call $\{x_1 \mapsto C_1, \ldots, x_k \mapsto C_k\}$ a candidate answer to Q if condition (1) in the above definition

³ To improve readability in the rest of the paper we allow constants to appear in CQs. However, conjuncts of the form x = a are just syntactic sugar for a conjunct $\{a\}(x)$ formed using a concept $\{a\}$. Similarly, f(x) = a is $\exists y(f(x) = y \land \{a\}(y))$, etc.

is satisfied, and call it a weakly identifying answer to Q if only an upper bound (of one) is guaranteed to hold in condition (2) We call concepts C_i that are used in this way singular referring concepts since they replace the role of individual names as referring expressions in certain answers to conjunctive queries.

To illustrate, assume \mathcal{K} captures information about persons, and consider a query with body $\operatorname{Person}(x)$. In the rest of the paper, we will use Pf = a as an alternative syntax for $\forall Pf.\{a\}$ (to improve readability). Possibilities for certain answers to the query now include one or more of the following:

$$\{x \mapsto (ssn\# = 1234)\}$$

 $\{x \mapsto Female \sqcap (hasSpouse.name = 'Enya') \sqcap (hasSpouse.phone \# = 1234567)\}.$ Note that Definition 5(2) disallows answers of the form $\{x \mapsto Female\}$ or more generally, rules out any C for x in which $|(C)^{\mathcal{I}}| \neq 1$ when $\mathcal{I} \models \mathcal{K}$. Thus, the earlier examples of certain answers would be contingent on $\mathcal{T}_{\mathcal{K}}$ ensuring that persons have unique ssn#, as well as unique spouses, who can be identified by a (name, phone #) pair.

4 Referring Concept Types in Conjunctive Queries

Allowing referring concepts beyond nominals in certain answers to a query Q may lead to infinitely many syntactically distinct certain answers. In this section, we develop a framework that ensures the set of certain answers to any conjunctive query is finite by introducing a specific language for *referring concept types*, which bottom out at individual nominals.

Important note: while Definition 5 allows general $C\mathcal{FD}_{nc}^{\forall}$ concepts to serve as *referring concepts*, in the rest of the paper we restrict our attention to the subset of referring concepts adhering to the more limited grammar

$$C ::= \{a\} \mid A \mid \forall Pf.C \mid C \sqcap C$$

where $\{a\}$ is a nominal, A is a primitive concept name, and $Pf \in F^*$. These C are intuitively instances of the following referring *types*.

Definition 6 (Referring Concept Types)

A referring concept type Rt is given by the following grammar, where T denotes a finite conjunction of primitive concepts (or \top standing for empty conjunction), to be called henceforth a *simple type*.

 $Rt ::= \{?\} \mid Pf = \{?\} \mid Rt_1 \sqcap Rt_2 \mid T \to Rt \mid Rt_1; Rt_2$ The referring concept set $RC(Rt, \mathcal{K})$ is the "extension" of a referring concept type Rt with respect to KB \mathcal{K} , and is defined inductively as follows, where S_i is

short for $\operatorname{RC}(Rt_i, \mathcal{K})$:

1. $\operatorname{RC}(\{?\}, \mathcal{K}) = \{\{a\} \mid a \text{ occurs in } \mathcal{A}_{\mathcal{K}}\};\$

2. $\operatorname{RC}(Pf = \{?\}, \mathcal{K}) = \{(Pf = \{a\}) \mid a \text{ occurs in ABox } \mathcal{A}_{\mathcal{K}}\};$

3. $\operatorname{RC}(Rt_1 \sqcap Rt_2, \mathcal{K}) = \{C_1 \sqcap C_2 \mid C_i \in S_i\};$

4. $\operatorname{RC}(T \to Rt_1, \mathcal{K}) = \{T \sqcap C \mid C \in S_1\};$ and

or

5. $\operatorname{RC}(Rt_1; Rt_2, \mathcal{K}) = S_1 \cup \{C_2 \in S_2 \mid \neg \exists C_1 \in S_1 \text{ s.t. } \mathcal{K} \models C_1 \equiv C_2\}.$

A referring concept type is *homogeneous* if it is free of any occurrence of the construct in 5. $\hfill \Box$

Examples of their use will follow immediately after the next definition. Note that, as desired, the set of referring concepts associated with a single referring concept type is finite if $\mathcal{A}_{\mathcal{K}}$ is finite.

Definition 7 (Certain Answers and Singular Referring Concepts)

Let Q be a conjunctive query with free variables $\{x_1, \ldots, x_k\}$. A query head H for Q is a set of pairs $\{x_1 : Rt_1, \ldots, x_k : Rt_k\}$ that associates a referring concept type Rt_i with each x_i .

The set of certain answers to Q with respect to a head H and a knowledge base \mathcal{K} , denoted ANS (Q, H, \mathcal{K}) , is the set of all certain answers $\{x_i \mapsto C_i \mid 0 < i \leq k\}$ to Q over \mathcal{K} for which $C_i \in \operatorname{RC}(Rt_i, \mathcal{K})$, for $0 < i \leq k$.

After the examples below, the objective in this section is to show that computing $\operatorname{ANS}(Q, H, \mathcal{K})$ can be achieved in PTIME for $\mathcal{CFD}_{nc}^{\forall}$, provided that referring concept types satisfy a "weak identification condition".

The following examples illustrate the use of referring concepts with conjunctive queries to extend the current situation with the more expressive cases motivated in the Introduction (a conjunctive query Q with a head H will be written in the following SQL-like style: select H where BODY(Q)):

1. (expressing the current case) "Any journals published by Italians" select $x_1 : \{?\}$

where $Journal(x_1) \land (publishedBy(x_1) = x_2) \land Italian(x_2)$

2. (reference via single key) "The ssn# of any person with phone 12345567" select $x : ssn\# = \{?\}$

where $Person(x) \land (phone \#(x) = 1234567)$

- 3. (multiple attribute key) "The title and publisher of any journals" select x : title = {?} □ publishedBy = {?} where Journal(x)
- 4. (choice of identification in heterogeneous set) "Any legal entity" select $x : Person \rightarrow ssn\# = \{?\}$; Company \rightarrow tickerSymbol = $\{?\}$ where LegalEntity(x)

An example certain answer would be $x \mapsto Person \sqcap (ssn\# = 7654)$ while another might be $x \mapsto Company \sqcap (tickerSymbol = 'IBM')$.

5. (preferred identification) "Any publication, identified by its most specific identifier, when available." select $x : Journal \rightarrow (title = \{?\} \sqcap publisher = \{?\});$

EditedCollection $\rightarrow isbn\# = \{?\}; \{?\}$ where Publication(x)

We now make concrete our requirement that referring concepts occurring in query answers do indeed satisfy the ability to identify objects. **Definition 8 (Weak Identification in Certain Answers)** Let Q be a conjunctive query, H a head for Q, and $\mathcal{T} \in C\mathcal{FD}_{nc}^{\forall}$ TBox. Q is weakly identifying for H with respect to \mathcal{T} if $|(C)^{\mathcal{I}}| \leq 1$ for every ABox \mathcal{A} , model \mathcal{I} of $\mathcal{T} \cup \mathcal{A}$, and candidate answer θ to Q with respect to H and $\mathcal{T} \cup \mathcal{A}$ in which $x \mapsto C \in \theta$. \Box

Lemma 9 (Normal Form of Referring Concept Types) For every referring concept type *Rt*, there is an *equivalent normal form*

$$Rt_1;\ldots;Rt_n,$$

denoted NORM(Rt), consisting of tagged record types Rt_i that are, in turn, homogeneous referring concept types of the form

$$T_i \to ((Pf_{i,1} = \{?\}) \sqcap \ldots \sqcap (Pf_{i,m_i} = \{?\})).$$
(4)

By equivalent, we mean that, for any KB \mathcal{K} , $C_1 \in \mathrm{RC}(Rt, \mathcal{K})$ implies there exists $C_2 \in \mathrm{RC}(\mathrm{NORM}(Rt), \mathcal{K})$ for which $\{\} \models C_1 \equiv C_2$, and vise versa. \Box

Proof (sketch): By application of the following equivalence preserving rewrites.

$$\{a\} \quad \rightsquigarrow \quad \{\} \rightarrow (id = \{a\})$$

$$(T \rightarrow Rt_1) \sqcap Rt_2 \text{ or } Rt_1 \sqcap (T \rightarrow Rt_2) \quad \rightsquigarrow \quad T \rightarrow (Rt_1 \sqcap Rt_2)$$

$$(Rp_1; Rt_2) \sqcap Rt_3 \quad \rightsquigarrow \quad (Rt_1 \sqcap Rt_3); (Rt_2 \sqcap Rt_3)$$

$$Rt_1 \sqcap (Rt_2; Rt_3) \quad \rightsquigarrow \quad (Rt_1 \sqcap Rt_2); (Rt_1 \sqcap Rt_3)$$

$$T \rightarrow (Rt_1; Rt_2) \quad \rightsquigarrow \quad (T \rightarrow Rt_1); (T \rightarrow Rt_2)$$

$$T_1 \rightarrow (T_2 \rightarrow Rt) \quad \rightsquigarrow \quad (T_1 \sqcap T_2) \rightarrow Rt \qquad \square$$

We now present our first main result on deciding weak identification. Our theorem refers to the following auxiliary function $\mathcal{M}(\cdot)$ which abstracts query bodies as DL concepts (note that the function assumes, without harm, that variables are also elements of F):

$$\mathcal{M}(\phi) = \begin{cases} \forall x.C & \text{if } \phi = ``C(x)"; \\ x_1.f = x_2 & \text{if } \phi = ``f(x_1) = x_2"; \\ \mathcal{M}(\psi_1) \sqcap \mathcal{M}(\psi_2) & \text{if } \phi = ``\psi_1 \land \psi_2"; \text{ and} \\ \mathcal{M}(\psi) & \text{otherwise, when } \phi = ``\exists x_{k+1}, \dots, \exists x_m : \psi". \end{cases}$$

Theorem 10 (Deciding Weak Identification) Let Q be a conjunctive query, H a head for Q, and $\mathcal{T} \in CFD_{nc}^{\forall}$ TBox. Q is weakly identifying for H with respect to \mathcal{T} if and only if

 $\mathcal{T} \cup \{A_Q \sqsubseteq ((\forall x.T) \sqcap \mathcal{M}(\text{BODY}(Q))) \models A_Q \sqsubseteq A_Q : x.Pf_{i,1}, \dots, x.Pf_{i,m_i} \to x \text{ for all } (x:Rt) \in H, \text{ and all } T \to (Pf_1 = \{?\}) \sqcap \dots \sqcap (Pf_m = \{?\}) \in \text{NORM}(Rt).$

<u>Proof (sketch)</u>: The if direction is straightforward. For the only-if direction, consider the earliest tagged record type $Rt' \in \text{NORM}(Rt)$ for some $x : Rt \in H$ for which the check for the logical consequence of the key PFD fails. Then one can construct an ABox \mathcal{A} where there exists $C \in \text{RC}(Rt', \mathcal{T} \cup \mathcal{A})$ that occurs in some candidate answer θ for Q for which one can also construct an interpretation \mathcal{I} for $\mathcal{T} \cup \mathcal{A}$ for which $|(C)^{\mathcal{I}}| > 1$.

Definition 11 Let Q be a CQ with free variables $\{x_1, \ldots, x_k\}$. We say that H is a homogeneous head for Q if it is of the form

 $H = \{ x_i : T_i \to (Pf_{i,1} = \{?\}) \sqcap \ldots \sqcap (Pf_{i,\ell_i} = \{?\}) \mid 0 < i \le k \}.$

For every CQ Q and a homogeneous head H for Q we define a conjunctive query

$$Q_H = Q \wedge \bigwedge_{i=1}^{\kappa} (T_i(x_i) \wedge (Pf_{i,1}(x_i) = x_{i,1} \wedge \ldots \wedge (Pf_{i,\ell_i} = x_{i,\ell_i}))$$

with additional free variables $\{x_{i,j} \mid 0 < i \le k, 0 < j \le \ell_i\}$.

To handle preferences, i.e., non-homogeneous heads H of queries of the form $\{x_i: Rp_{i,1}; \ldots; Rp_{i,n_i} \mid 0 < i \leq k\}$ we first define a sequence of homogeneous heads

$$H_{j_1,\dots,j_k} = \{x_i : Rp_{i,j_i} \mid 0 < i \le k\}$$

over all $0 < j_i < n_i$. In addition, we define a *normalized* ABox \mathcal{A}' for an ABox \mathcal{A} to contain only individual relationship assertions of the form f(a) = b obtained from \mathcal{A} by introducing additional individuals for the intermediate individuals participating in \mathcal{A} 's individual relationship assertions.⁴

Theorem 12 Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, Q a CQ, and H a head for Q, such that Q is weakly identifying for H in \mathcal{T} . Then

 $\{x_i\mapsto T_i\,\sqcap\,(Pf_{i,1}=\{a_{i,1}\})\,\sqcap\,\ldots\,\sqcap\,(Pf_{i,\ell_i}=\{a_{i,\ell_i}\})\}\in \operatorname{Ans}(Q,H,\mathcal{K})$ if and only if

 $\{x_i \mapsto \{b_i\}, x_{i,j} \mapsto \{a_{i,j}\} \mid 0 < i \le k, 0 < j \le \ell_i\} \in \operatorname{Ans}(Q_{H_{j_1,\dots,j_k}}, H_0, \mathcal{K}')$ and there is no

 $\{x_i \mapsto \{b_i\}, x_{i,j} \mapsto \{a_{i,j}'\} \mid 0 < i \le k, 0 < j \le \ell_i\} \in \operatorname{Ans}(Q_{H_{j_1', \dots, j_k'}}, H_0, \mathcal{K}')$

for $H_{j'_1,\ldots,j'_k}$ dominates H_{j_1,\ldots,j_k} , where $H_0 = \{x_i : \{?\} \mid 0 < i \le k\} \cup \{x_{i,j} : \{?\} \mid 0 < i \le k, 0 < j \le \ell_i\}, \mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ where \mathcal{A}' is a normalized \mathcal{A} , and $a_{i,j}$ and $a'_{i,j}$ are constants in \mathcal{A} .

Proof (sketch): Consider first the case where H is homogeneous. Then we can reconstruct an answer to Q and H from an answer to Q_H and H_0 (which consists of nominal concepts only. Conversely, from answer to Q and H we can extract nominal concepts that are an answer to $\exists x_1, \ldots, x_k.Q_H$ and H_0 (restricted to free variables of $\exists x_1, \ldots, x_k.Q_H$). This answer is then extended to Q_H and (full) H_0 by using individual names introduced in the normalized ABox \mathcal{A}' , this extension is unique since Q is weakly identifying for H in \mathcal{T} .

For the non-homogeneous case we simply consider all possible homogeneous subcases and then filter the answers by the valuations of the variables x_1, \ldots, x_k (since those describe the possibly anonymous individuals in \mathcal{A} using their systemassigned names A').

Note that the queries $Q_{H_{j'_1,\ldots,j'_k}}$ are answered in \mathcal{K} with respect to H_0 , a trivial and homogeneous head: this reduces the answering to standard CQ query answering in $\mathcal{CFD}_{nc}^{\forall}$ knowledge base \mathcal{K} as introduced in [12].

Corollary 13 Computing certain answers to conjunctive queries with respect to referring concepts and $\mathcal{CFD}_{nc}^{\forall}$ knowledge bases is complete for PTIME.

⁴ Such a transformation is already part of the CQ answering algorithm [12]. Note that in our setting, however, the new individuals cannot participate in Q's answers.

5 Conclusions

The paper's contributions are as follows.

First and foremost, on the non-technical side, it recognized and motivated the utility of "singular referring expressions" for 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 does) vs. identification (how results are presented).

On the specification side, the paper defined formally the notion of "query answering using referring expressions" for certain answers in conjunctive queries over DLs. In the context of $\mathcal{CFD}_{nc}^{\forall}$, it introduced a specific language for referring expressions, which are a subset of $\mathcal{CFD}_{nc}^{\forall}$ concepts, and which allows us to handle all the motivating problems (except intensional descriptions in this paper. This language generalizes the notion of nominal, currently used in OBDA, to handle keys (as found in both the database and DL KB literature), and supports heterogeneous sets (as in the case of *LegalEntity*), as well as preferential choice of referring expressions (as in the *EditedCollection* example).

On the algorithmic and complexity side, it first considered, in the context of $\mathcal{CFD}_{nc}^{\forall}$, the problem of determining (in polynomial time) whether a referring concept type was "weakly" identifying in the context of a query and TBox, in the sense that its instances necessarily referred to at most one object. It also showed how one can transform a query and knowledge base so that the answers had cardinality one. This led to the result that computing certain answers to conjunctive queries with respect to referring concepts and $\mathcal{CFD}_{nc}^{\forall}$ KBs was complete for PTIME.

There are many avenues left to explore in this work. We have already mentioned that lack of space prevented us from considering referring concepts types for "Michelle's mother", which use inverse functions. Another direction to consider are additional forms or desirable properties of referring expressions. For example, the variety of references raises a problem: the same object may be returned in an answer with different references to it, even if the knowledge base works with the UNA. As a simplest example, a journal has multiple candidate keys. We are currently investigating ways to reason about and avoid such duplication.

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.

Of course, all the technical questions considered in this paper will have different answers for different DLs. Therefore, an orthogonal avenue of research is to re-consider these issues in the context of other DLs, especially DL-Lite.

Acknowledgments: We wish to thank those reviewers who have made excellent suggestions for improving the paper, and financial support from NSERC Canada.

References

- Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. 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, page 3 pages, 1995.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In ACM SIGMOD International Conference on Management of Data, pages 1247–1250. ACM, 2008.
- Alexander Borgida. Description logics in data management. Knowledge and Data Engineering, IEEE Transactions on, 7(5):671–682, 1995.
- Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. Semantic Web, 2(1):43–53, 2011.
- 5. Ramez Elmasri and Shamkant B. Navathe. Fundamentals of Database Systems, 3rd Edition. Addison-Wesley-Longman, 2000.
- Tomasz Imielinski. Intelligent query answering in rule based systems. The Journal of Logic Programming, 4(3):229–257, 1987.
- Andrey Andreyevich Markov, Jr. On the impossibility of certain algorithm in the theory of associative systems. *Dokl. Akad. Nauk SSSR*, 55:587–590, 1947.
- Amihai Motro. Intensional answers to database queries. Knowledge and Data Engineering, IEEE Transactions on, 6(3):444–454, 1994.
- Emil Post. Recursive unsolvability of a problem of Thue. The Journal of Symbolic Logic, 12:1–11, 1947.
- Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. Database System Concepts, 4th Edition. McGraw-Hill Book Company, 2005.
- David Toman and Grant E. Weddell. Conjunctive query answering in CFD_{nc}: A PTIME description logic with functional constraints and disjointness. In AI 2013: Advances in Artificial Intelligence - 26th Australasian Joint Conference, Dunedin, New Zealand, December 1-6, 2013. Proceedings, pages 350–361, 2013.
- 12. David Toman and Grant E. Weddell. Answering Queries over $\mathcal{CFD}_{nc}^{\forall}$ Knowledge Bases. Technical Report CS-2014-14, Cheriton School of Computer Science, University of Waterloo, 2014.
- David Toman and Grant E. Weddell. On adding inverse features to the description logic CFD[∀]_{nc}. In PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014., pages 587–599, 2014.