Concerning Referring Expressions in Query Answers *

Alexander Borgida†, David Toman‡ and Grant Weddell‡
†Department of Computer Science, Rutgers University, New Brunswick, USA
‡Cheriton School of Computer Science, University of Waterloo, Canada
†borgida@cs.rutgers.edu, ‡{david,gweddell}@uwaterloo.ca

Abstract

A referring expression in linguistics is a noun phrase that identifies individuals to listeners. In the context of a query over a first order knowledge base, referring expressions to answers are usually constant symbols. This paper motivates and initiates the exploration of allowing more general formulas, called singular referring expressions, to replace constants in this role. Referring expression types play a novel and significant role in analyzing the properties of candidate expressions.

1. Introduction and Motivation

Query answering in logic-based approaches to data and knowledge bases has traditionally involved finding constant names which can be substituted for the free variables of the query in such a way that the knowledge-base (KB) entails it. In a number of circumstances this is less than ideal:

(1) In object-centered KBMSs (including Object-Oriented DBMSs, and KRR systems such as Description Logics (DLs) with Unique Name Assumption), all known individual objects must have unique (internal) distinguishing identifiers. However, these identifiers are often insufficient to allow humans to figure out what real-world object they refer to, especially in 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 was "/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 uniquely identify rows of tables. Exceptionless keys are however frequently impossible to find (e.g., newly arrived foreign students do not have ssn#), and the problem gets worse when databases are merged. For this reason, many tables use as keys columns of meaningless integers (often labeled with suffix _ID as in emp_ID), supported by the “auto-increment” feature available in modern RDBMS. Of course, such identifiers also obscure the meaning of answers.

(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, this 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 could 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 is a natural preference for journals to be described using the more meaningful (title, publisher) pairs over isbn# for edited collections.

In connection with (3) and (4), we note that Halpin [Halpin, 2015] has independently investigated many examples of complex reference schemes, and how they are (not) handled in some conceptual modeling languages.

(5) Many kinds of KBMSs allow one to describe situations where objects can be inferred to exist, without having an explicit (internal) identifier. For example, if Fred is a married person, then he has a mother and a spouse, who are women. Normally, such objects cannot be returned in the list of answers unless they are named constants. So they might not be returned by a query asking for all women. Yet it is common in English to identify objects by their relationship to other known objects. For example, “Fred’s mother” is a perfectly reasonable referring expressions to a woman.

This paper proposes a first solution to (some of) the issues raised by providing singular referring expressions in place of individual names returned by queries, in the context of FOL KBs, and then briefly applies this to conjunctive queries for some “lightweight” tractable dialects of DLs.

*This paper was invited for submission to the Best Papers From Sister Conferences Track, based on a paper that appeared in KR 2016 [Borgida, Toman, and Weddell, 2016a].
Since a referring expression is a noun phrase that picks out some individuals, it can naturally be seen as a formula $\phi$, with one free variable, denoting those individuals for which $\phi$ is true. Note that the current case of referring to an object by its name, table 1 say, can then be accomplished by the formula $(x = \text{table 1})$.

We start by proposing a language for referring expressions (REs) – one of many possible – and consider formally what it means to answer a query by returning REs. We want finite answers, yet there can be an unbounded number of REs referring to an individual. So we introduce a second language of referring expression types (RETs), and annotate each query variable with a RET, which limits the REs returned. Because we generalize the usual case of constant names in answers, we need a third property; the REs returned need to be singular: i.e., ones which always uniquely identify one individual.1

Unfortunately, it is impossible, for example, to tell in general whether an expression such as “object with $p$-value 3” will be guaranteed singular or not: if $p$ is a key, then yes; otherwise no. Therefore, we need to use general knowledge from the KB to verify at “compile time” 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).

2. Referring Expressions and Certain Answers

To be precise, we assume a KB 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. The standard Tarskian semantics is also assumed.

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

Currently, the space of possible answers to $\psi$ over $K$, a given KB, corresponds to substitutions $(x_1/a_1, \ldots, x_k/a_k)$, hence written as $\{x_i \mapsto a_i, \ldots, x_k \mapsto a_k\}$, that map each free variable of $\psi$ to a constant symbol. Also, recall that $(x_1/a_1, \ldots, x_k/a_k)$ qualifies as a certain answer when $K \models \psi(x_1/a_1, \ldots, x_k/a_k)$. The following definition captures precisely the notion singular referring expressions as answers.

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

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

We write $C_\theta$ as shorthand for $\bigwedge_{1 \leq i \leq k} \phi_i(x_i)$ (where the underlying query will be clear from context).

Let $K$ be a consistent KB. Then $\theta$ is a certain RE-answer to query $\psi$ with respect to $K$ if it satisfies the following two conditions:

$$K \models \exists x_1, \ldots, x_k : (C_\theta \land \psi)$$ and

$$K \models \forall x_1, \ldots, x_k : (C_\theta \Rightarrow \psi).$$ (2)

$\theta$ is said to be singular if it is a certain RE-answer and satisfies a third condition for each $1 \leq j \leq k$:

$$K \models \forall x_1, \ldots, x_{j-1}, y_1, y_2, x_{j+1}, \ldots, x_k : ((C_\theta \land \psi)(x_j/y_1) \land (C_\theta \land \psi)(x_j/y_2)) \Rightarrow (y_1 = y_2). \Box$$

Complex referring expressions might be $\text{hasAge}(x_1, 25)$ and $\exists y: \text{hasFather}(x_1, y) \land \text{hasLastName}(y, "Castor")$, 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”.

A possible answer/RE-substitution is really just a syntactic notion. Conditions (2) and (3) tie it to the meaning of answering query $\psi$ in knowledge base $K$; in particular, (3) disallows those RE-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 $\exists x_1: \phi(x_1)$ is false for some $i$.

Thus, consider a query $\psi\{x\}$ and RE-substitution $\{x \mapsto \text{Person}(x)\}$. If $K$ contains the axiom $\forall x: (\text{Person}(x) \Rightarrow \psi(x))$, then all persons are answers to the query in the context of $K$, since the axiom is (3). If (2), i.e., the sentence $K \models \exists x: \text{Person}(x) \land \psi$ is also true in $K$, then there is at least one answer of this form. Observe in this case, however, that $\text{Person}(x)$ is not likely to satisfy the final singularity condition on certain RE-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 RE-answers, and will also allow us to characterize compile-time identification properties of such typed queries with respect to knowledge bases corresponding to an ontology or a mediated database schema, that is, knowledge bases $K$ in which ground facts are absent. For the remainder of the paper, we write $K$ to refer to ontologies, and write $K \cup D$ when referring to a full knowledge base consisting of an ontological component $K$ and a data component $D$. Note that in addition to ground facts, the data component $D$ may contain other kinds of knowledge, for example, closure axioms when the CWA [Reiter, 1984] is desired.

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 $\text{Person} \rightarrow (\text{name}.last = \{?\} \land \text{phone#} = \{?\})$ indicates that an object $x$ in class $\text{Person}$ may be identified by a pair of constants $s$ and $p$ such that $\exists y. (\text{name}(x, y) \land \text{last}(y, s) \land \text{phone#}(x, p))$. A binary preference operator “;” is also included as a means for indicating user preferences among referring expressions. Formally:
Definition 2 (Referring Expression Types) Let \( Pd \) stand for a path constructed by composition of binary relations, and \( T \) be a unary formula (called a guard). A referring expression type \( Rt \) is generated by the following production

\[
Rt ::= Pd = \{?\} \mid Rt_1 \circ Rt_2 \mid T \rightarrow Rt \mid Rt_1; Rt_2
\]

We define the semantics of \( Rt \) with respect to a knowledge base \( K \). In particular, we write \( \text{RE}(Rt,K) \) to associate a set of referring expressions with the free variable \( x \) to each \( Rt \), in the context of \( K \). \( \text{RE}(Rt,K) \) is defined inductively as follows:

1. \( \text{RE}(Pd = \{?\},K) = \{\exists y : (Pd(x,y) \land y = b) \mid b \text{ a constant symbol} \} \);
2. \( \text{RE}(Rt_1 \circ Rt_2,K) = \{\phi_1 \land \phi_2 \mid \phi_1 \in S_1 \land \phi_2 \in S_2\}; \)
3. \( \text{RE}(T \rightarrow Rt_1,K) = \{T(x) \land \phi \mid \phi \in S_1\}; \)
4. \( \text{RE}(Rt_1; Rt_2,K) = S_1 \cup \{\phi_2 \in S_2 \mid \neg \exists \phi_1 \in S_1 \text{ s.t. } K \models \forall x : (\phi_1 \equiv \phi_2)\}. \)

where \( S_i \) is shorthand for \( \text{RE}(Rt_i,K) \).

Referring expression types allow us to annotate query variables as follows:

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

\[
\{x_1 : Rt_1, \ldots , x_k : Rt_k\}.
\]

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

We now illustrate the use of referring expression types in typed queries to resolve the issues raised in the introduction. In the following, each query \( \psi \) with head \( \text{HD}(\psi) \) will be written in an SQL-like style "select \( \text{HD}(\psi) \) where \( \psi \):

Q.1 (expressing the current case) “Find places at which people live”

select \( x_2 : \{?\} \)
where \( \exists x_1 : \text{Person}(x_1) \land \text{livesAt}(x_1, x_2) \)

Q.2 (reference via single key) “Find persons having age 35 (identified by ssn#)”

select \( x : \text{ssn#} = \{?\} \)
where \( \text{Person}(x) \land \text{hasAge}(x, 35) \)

Q.3 (multiple attribute key) “Find journals (identified by their title and publisher)”

select \( x : \text{title} = \{?\} \land \text{publishedBy} = \{?\} \)
where \( \text{Journal}(x) \)

Q.4 (choice of identification in heterogeneous set) “Find legal entities”

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

Possible certain answers for this query might be:

\[
\{x \mapsto \text{Person}(x) \land \text{ssn#}(x, 7654)\}
\{x \mapsto \text{Company}(x) \land \text{tickerSymbol}(x, \text{"IBM"})\}
\]

Q.5 (preference in identification) “Find publications (identified by their most specific identifier, when available).”

select \( x : \text{Journal} \rightarrow (\text{title} = \{?\} \land \text{publishedBy} = \{?\}); \text{EditedCollection} \rightarrow \text{isbn#} = \{?\} ; \{?\} \)
where \( \text{Publication}(x) \)

For this query, certain answers might include:

\[
\{x \mapsto \text{Journal}(x) \land \text{title}(x, \text{"AIJ")} \land \\
\text{publisher}(x, \text{"Elsevier")}; \}
\{x \mapsto \text{EditedCollection}(x) \land \text{isbn#}(x, 123456789)\}
\{x \mapsto x = /\text{guid}/9202a8c04000600...\}
\]

Q.6 (multiple forms of reference, including intensional) “Find women”

select \( x : \{?\}; \text{spouseOf} = \{?\}; \text{motherOf} = \{?\} \)
where \( \text{Person}(x) \land \text{hasGender}(x, \text{‘female’}) \)

In the final case, a certain answer such as

\[
\{x \mapsto \text{motherOf} = \text{Fred}\}
\]

is a particularly good illustration of a referring expression describing an individual whose identity is not known, but can be inferred to exist and have the property of being a woman; such answers 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 and publishedBy values in query (3), and so on. We formally characterize this requirement in terms of the ontological component \( K \) 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 \( K \) be a consistent knowledge base and \( \psi \) a typed query. Then \( \text{HD}(\psi) \) is weakly identifying for \( \psi \) with respect to \( K \) iff, for all \( D \) consistent with \( K \), every certain answer \( \theta \) of \( \psi \) with respect to \( K \cup D \) is singular.

[Borgida, Toman, and Weddell, 2016a] show that the problem of deciding weak identification for an ontology given by a consistent FO knowledge base and a typed FO query reduces to a logical implication problem in the underlying logic.

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 answer, regardless of the data component of a knowledge base, many KBMSs assume an additional condition on constant symbols, the so-called unique name assumption (UNA), which stipulates that distinct symbols denote distinct individuals. A natural generalization to referring expressions, called strong identification, analogously requires syntactically distinct REs to
be disjoint with respect to the KB (singularity then guarantees they refer to a unique object). [Borgida, Toman, and Weddell, 2016a] has developed sufficient conditions to test for this property for a given typed query with respect to the ontological component of a KB.

3. Query Answering with Referring Expressions

We now present a characterization of query answering in our framework in terms of an oracle. In particular, given a query $\psi\{x_1, \ldots, x_k\}$, and (full) knowledge base $K \cup D$, we assume that the oracle can provide an iterator over all tuples of constants $a_1, \ldots, a_k$ such that $K \cup D \models \psi(x_1/a_1, \ldots, x_k/a_k)$. Query answering with referring expressions then uses the oracle as follows:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Normalize $H$ to $H_1; \ldots; H_\ell$ with each $H_i$ of the form $T_i \rightarrow Pd_{i,1}={?} \land \cdots \land Pd_{i,\ell}={?}$;</td>
</tr>
<tr>
<td>2.</td>
<td>Let queries $\psi_i{x, y_1, \ldots, y_k}$ stand for $\psi \land T_i(x) \land Pd_{i,1}(x, y_1) \land \cdots \land Pd_{i,\ell}(x, y_k)$;</td>
</tr>
<tr>
<td>3.</td>
<td>Create $D_i$ with a witness for $x$, when no such witness exists in $D$; (see below)</td>
</tr>
<tr>
<td>4.</td>
<td>Evaluate $\psi_i$ in $K \cup D \cup D_i$ (using the oracle);</td>
</tr>
<tr>
<td>5.</td>
<td>Resolve preferences (based on value of $x$); and</td>
</tr>
<tr>
<td>6.</td>
<td>Reconstruct a referring expression from the constant symbols retrieved for $y_1, \ldots, y_k$.</td>
</tr>
</tbody>
</table>

Intuitively, the algorithm considers each preference among referring expressions stipulated by $H$ separately: step (1) separates the preferences into separate preference-free heads $H_i$. Each of these is then transformed to a conjunctive query that is conjoined to the user query $\psi$ in step (2). Evaluating this query with respect to $K \cup D$ may, however, miss answers as the individual for $x$ could be anonymous. Step (3) temporarily creates additional auxiliary constants (called witnesses) to avoid this case. One can always do this since the referring expression types in $H$ are weakly identifying. Next, we use the oracle to find (standard) answers to the modified queries in step (4), resolve preferences by selecting the most preferred answer among those with the same value for $x$ in step (5), and then use the corresponding values of $y_i$’s to reconstruct the most preferred referring expression describing that particular individual.

**Theorem 5** Let $K \cup D$ be a consistent knowledge base and $\psi\{x_1, \ldots, x_n\}$ a normalized typed query such that $HD(\psi)$ is weakly identifying for $\psi$ in $K$. Then the above procedure correctly computes all RE-substitutions for $\psi$ w.r.t. $K \cup D$.

The Case of Description Logics.

Concepts in a description logic correspond to unary predicates and are therefore natural candidates to serve as referring expressions, as observed by Computational Linguists (e.g., [Areces, Figueira, and Gorin, 2011]). Our framework therefore applies very naturally to ontology-based data access (OBDA), where ontologies are KBs expressed in a DL, and where one usually considers conjunctive queries. Referring expressions in answers would now be given in a variable-free form as concepts; for example

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

and

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

Specific instances of tractable oracles are the query answering approaches using “perfect rewriting” for ontologies formulated in the logic DL-Lite$_{core,\{sic\}}$ [Calvanese et al., 2007; 2008], and ones using the “combined approach” in the logic CFD$_{\alpha\beta\gamma}$ [Toman and Weddell, 2013; St. Jacques, Toman, and Weddell, 2016]. Note that weak identification can be reduced to knowledge base consistency for both of these logics, which are also tractable problems. Combining the OBDA procedures for the above logics leads to tractable OBDA extensions for referring expressions. Further details on how one specializes the REs and their types to these logics can be found in [Borgida, Toman, and Weddell, 2016a].

We note that in the case of DLs and other logics that enjoy the tree model property, step (3) in the above outline can be avoided using additional query reformulation, as shown in [Toman and Weddell, 2016].

4. Discussion

The question of finding a referring expression for a set of desired individuals is a central one in Computational Linguistics. Krahmer and van Deemter [Krahmer and van Deemter, 2012] provide an extensive survey of approaches to generating referring expressions. Some authors [Areces, Figueira, and Gorin, 2011] have in fact considered the issue of using different kinds of logical languages, including FOL and DLs, to describe references. This field differs from our work because it assumes a single model of the world. More importantly, our focus on typed queries and compile-time analysis for singular expressions differentiates our work from the direction previously pursued.

Overall, this paper recognized and motivated the utility of “singular referring expressions” (formulas with a single free variable) as query answers, which are more complex than simple constant symbols, and it argued for a separation of concerns in query writing: identifying the objects to be returned vs describing them for users. The paper defined formally the notion of “query answering from a knowledge base using singular referring expressions”; introducing a first language for referring expressions and their types (which make finite the set of referring expressions to consider as answers).

Another use of referring expression types allows one to resolve how objects are identified in a given data source or knowledge base without resorting to artificially generated identifiers [Borgida, Toman, and Weddell, 2016b]. This is achieved by associating such types with primitive concepts and provides for a separation of conceptual modelling from identification concerns.
References


