Identity Resolution in Conjunctive Querying over DL-based Knowledge Bases

David Toman and Grant Weddell

Cheriton School of Computer Science, University of Waterloo, Canada
{david,gweddell}@uwaterloo.ca

Abstract. Earlier work has proposed a notion of referring expressions and types in first order knowledge bases as a way of more effectively answering conjunctive queries in ontology based data access (OBDA). We consider how PTIME description logics can be combined with referring expressions to provide a more effective virtual front-end to relational data sources via OBDA. In particular, we consider replacing the standard notion of an assertion box, or ABox, with a more general notion of a concept box, or CBox.

1 Introduction

In a query answer $(a_1, \ldots, a_n)$ over a first order knowledge base $\mathcal{K}$, the common assumption is that each a_i will correspond to some constant symbol occurring in $\mathcal{K}$. A more general option has been proposed in [1] in which each a_i can now be a *referring expression*, in particular, a well-formed formulae ψ that is free in one variable and that satisfies a number of additional conditions for any interpretation $\mathcal{I}$ of $\mathcal{K}$. First, ψ should not be *vacuous*: it should hold of at least one individual in $\triangle^{\mathcal{I}}$. Second, ψ should be *singular*: it should hold of *at most one* individual in $\triangle^{\mathcal{I}}$. And third, the singularity property of ψ should be ensured by the ontological component of $\mathcal{K}$.

In this paper, we consider query answering in which the ontological component of $\mathcal{K}$ consists of a TBox $\mathcal{T}$ expressed in terms of a DL, and in which the remaining part of $\mathcal{K}$ consists of a CBox $\mathcal{C}$ instead of an ABox, where $\mathcal{C}$ consists of a finite set of referring expressions in the form of concept descriptions in the DL, and for which each is presumed non-empty and singular in all models of $\mathcal{K}$.

The DL we consider is *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ [4, 6], a dialect of the PTIME feature-based $\mathcal{CFD}$ family designed for capturing relational data sources, and our main focus is on query answering over a knowledge base $\mathcal{K}$ consisting of a TBox and CBox pair $(\mathcal{T}, \mathcal{C})$ expressed in terms of *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$. The main technical difficulty is on mapping $\mathcal{C}$ to a combination of an ABox $\mathcal{A}$ and a way of distinguishing the constant symbols occurring in $\mathcal{A}$ that "stand in place" of referring expressions in $\mathcal{C}$. This must be done in a way that ensures off-the-shelf query answering over $(\mathcal{T}, \mathcal{A})$ can be used to compute the certain answers to queries over the original $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ by a simple substitution of the distinguished constants by their referring expressions.

A core problem in deriving the ABox relates to identity issues when introducing new constants. Of particular significance is that fix-point computations are necessary when such constants are introduced. Indeed, this can be necessary when a TBox derives from relational data sources with tables that have uniqueness constraints as well as primary keys, or for which primary keys themselves are not minimal. In database parlance, one would say in this case that primary keys are *superkeys* but not *candidate keys*. Such "key conversion" tables can serve to map between alternative primary keys and thereby lead to additional query answers.

While several approaches to integrating information in settings in which the same individual can be identified in several (even syntactically incomparable) ways have been considered in the past [2], we show how the integration can be achieved naturally within $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$ by reducing the problem to existing ABox completion procedures for query answering over $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$. In particular, using concepts and procedures developed in [1] and [4], we define a natural way of capturing (perhaps multiple) external identities of objects. Subsequently, we show how query answering can be achieved in such a setting via an embedding into standard $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$ reasoning and query answering problems. We also present examples that show applications of this technique both in the relational setting and in the setting of document databases such as MongoDB.

The remainder of the paper is organized as follows. We begin with the necessary background material in Section 2 in which we introduce $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$ concepts, and "standard" knowledge bases consisting of a TBox of inclusion dependencies over such concepts and an ABox of assertions. Our main results then follow in Section 3 in which an ABox is replaced with a CBox of $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$ concepts called *referring expressions*. We then define a mapping of CBoxes to ABoxes and show how each of the following can be resolved with the use of this mapping: (1) diagnosing an *admissibility* condition for a CBox, (2) satisfiability of knowledge bases with a CBox, and (3) query answering over knowledge bases with a CBox. The admissibility condition requires that the TBox ensures each referring expression occurring in the CBox is singular in the sense outlined above. Throughout, we introduce examples to illustrate why CBoxes are useful and how identification issues become far more complicated as a consequence. We conclude with summary comments in Section 4.

2 Background

The description logic $\mathit{partial}{-}\mathcal{CFDI}_{nc}^{\forall-}$ is a member of the $\mathcal{CFD}$ family of DLs which 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 *features*. Note that features deviate from the normal practice of admitting *roles* denoting binary predicate symbols. However, features make it easier to incorporate concept constructors that are better suited to the capture of relational data sources, particularly so when they

include dependencies such as primary keys, uniqueness constraints, functional dependencies and foreign keys. This is achieved by a straightforward reification of n-ary predicates and by using a concept constructor peculiar to the $\mathcal{CFD}$ family called a *path functional dependency*. Consider the case of a role R. It can be reified as a primitive concept R_C, two features $domR$ and $ranR$ and an inclusion dependency of the form

$$R_C \sqsubseteq R_C : domR, ranR \rightarrow id$$

in *partial*$-\mathcal{CFD}\mathcal{I}_{nc}^{\forall-}$. The latter effectively ensures any combination of $domR$ and $ranR$ values uniquely determine an R 2-tuple. Note that an $\mathcal{ALC}$ inclusion dependency mentioning R of the form "A $\sqsubseteq \forall R$.B", can also be captured in *partial*$-\mathcal{CFD}\mathcal{I}_{nc}^{\forall-}$ as the inclusion dependency

$$\forall domR.\mathrm{A} \sqsubseteq \forall ranR.\mathrm{B}.$$

Concepts in *partial*$-\mathcal{CFD}\mathcal{I}_{nc}^{\forall-}$ are defined as follows:

Definition 1 (*partial*$-\mathcal{CFD}\mathcal{I}_{nc}^{\forall-}$ Concepts)
Let F and PC be sets of feature names and primitive concept names, respectively. A *partial path expression* is defined by the grammar "$\mathsf{Pf} ::= f.\mathsf{Pf} \mid id$" for $f \in \mathsf{F}$. We define derived *concept descriptions* by the grammar on the left-hand-side of Fig. 1. A *path functional dependency* concept, or PFD, is obtained by using the final production of this grammar.

An *inclusion dependency* $\mathcal{C}$ is an expression of the form $C_1 \sqsubseteq C_2$. A *terminology* (TBox) $\mathcal{T}$ consists of a finite set of inclusion dependencies. A *posed question* $\mathcal{Q}$ is a single inclusion dependency.

Syntax	Semantics: Defn of "$\cdot^{\mathcal{I}}$"	
$C ::= \mathrm{A}$	$\mathrm{A}^{\mathcal{I}} \subseteq \triangle$	(primitive concept; $\mathrm{A} \in \mathsf{PC}$)
$\mid C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(conjunction)
$\mid \neg C$	$\triangle \setminus C^{\mathcal{I}}$	(negation)
$\mid \forall \mathsf{Pf}.C$	$\{x : \mathsf{Pf}^{\mathcal{I}}(x) \in C^{\mathcal{I}}\}$	(value restriction)
$\mid \exists \mathsf{Pf}$	$\{x : \mathsf{Pf}^{\mathcal{I}}(x)$ exists$\}$	(existential restriction)
$\mid \exists f^{-1}.C$	$\{f^{\mathcal{I}}(x) : x \in C^{\mathcal{I}}\}$	(inverse feature)
$\mid \{a\}$	$\{a^{\mathcal{I}}\}$	(nominal)
$\mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0$	(*see text*)	(PFD)

Fig. 1. Syntax and semantics of $\mathcal{CFD}$ concepts.

The *semantics* of expressions is defined with respect to a structure $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $\cdot^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concepts A to be subsets of $\triangle$ and primitive features f to be partial functions $f^{\mathcal{I}} : \triangle \rightarrow \triangle$. Note that *partial*$-\mathcal{CFD}\mathcal{I}_{nc}^{\forall-}$ adopts the

strict interpretation of undefined values, which means that arguments terms *must* be defined whenever equality and set membership do hold.

The interpretation is extended to partial path expressions, $id^{\mathcal{I}} = \lambda x.x$, $(f.\,\mathsf{Pf})^{\mathcal{I}} = \mathsf{Pf}^{\mathcal{I}} \circ f^{\mathcal{I}}$, in the natural way, and derived concept descriptions C not including PFDs as defined in the centre column of Fig. 1. For concept descriptions that are PFDs, the interpretation is defined as follows:[1]

$$(C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}_0)^{\mathcal{I}} = \{x \mid \forall y.y \in C^{\mathcal{I}} \wedge x \in (\exists \mathsf{Pf}_0)^{\mathcal{I}} \wedge y \in (\exists \mathsf{Pf}_0)^{\mathcal{I}} \wedge \bigwedge_{i=1}^{k} (x \in (\exists \mathsf{Pf}_i)^{\mathcal{I}} \wedge y \in (\exists \mathsf{Pf}_i)^{\mathcal{I}} \wedge \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y)) \to \mathsf{Pf}_0^{\mathcal{I}}(x) = \mathsf{Pf}_0^{\mathcal{I}}(y)\}.$$

An interpretation $\mathcal{I}$ *satisfies an inclusion dependency* $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ and is a *model of* $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) if it satisfies all inclusion dependencies in $\mathcal{T}$. The *logical implication problem* asks if $\mathcal{T} \models \mathcal{Q}$ holds, that is, if $\mathcal{Q}$ is satisfied in all models of $\mathcal{T}$. □

Observe that features are still *functional*, and that there is therefore no need for a qualified existential restriction of the form $\exists f.C$, since such restrictions can be equivalently written as $\forall f.C \sqcap \exists f$. Hence the use of qualified existential restrictions in the rest of the paper should be considered to be syntactic sugar.

To ensure PTIME reasoning in *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ we require all subsumptions in the TBox to adhere to the following restrictions (for more general TBoxes and normalization see [6]): all subsumptions have to be of the form $C \sqsubseteq D$, where the structure of concepts C and D are given by the following grammars:

$$C ::= A \mid \forall f.A$$
$$D ::= A \mid \bot \mid \neg A \mid \forall f.A \mid \exists f^{-1} \mid \exists f \mid A : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$$

In addition, PFDs must adhere to one of the following two forms to avoid undecidability [5]:

$$\begin{array}{l} 1.\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.\,\mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \to \mathsf{Pf} \ \text{ or} \\ 2.\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.g, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}\,.f \end{array} \tag{1}$$

Definition 2 (*partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ ABox)
The second component of a *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base is an ABox $\mathcal{A}$ that contains assertions of the form "A(a)", "$a = b$", "$a \neq b$", and "$f(a) = b$", with the usual interpretation mapping constant symbols to domain elements, and interpreting the assertions as set membership and an equality/inequality between a constant and another constant or a function application to a constant, respectively. □

Proposition 3 (*partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ KB Satisfiability[6])
Satisfiability of *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases is complete for PTIME.

[1] This constitutes the minimum necessary conditions needed for recognizing when one violates an inclusion dependency of the form "$C_1 \sqsubseteq C_2 : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}_0$".

Conjunctive queries are, as usual, formed from atomic queries (or *atoms*), corresponding to concept descriptions, and equalities between variables and application of function to variables, using conjunction and existential quantification. To simplify notation, we conflate conjunctive queries with the set of its constituent atoms and a set of *answer variables*:

Definition 4 (Conjunctive Query)
Let φ be a set of atoms (representing a conjunction) $A(x_i)$ and $f(x_{i_1}) = x_{i_2}$, where A is a primitive concept description, f a feature (including *id*), and $\bar{x}$ a tuple of variables. We call the expression $\{\bar{x} \mid \varphi\}$ a *conjunctive query* (CQ). □

A conjunctive query $\{\bar{x} \mid \varphi\}$ is therefore a notational variant of the formula $\exists \bar{y}. \bigwedge_{\psi \in \varphi} \psi$ in which $\bar{y}$ contains all variables appearing in φ but not in $\bar{x}$. The usual definition of certain answers is given by the following:

Definition 5 (Certain Answer)
Let $\mathcal{K}$ be a *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ KB and $Q = \{\bar{x} \mid \varphi\}$ a CQ. A *certain answer* to Q over $\mathcal{K}$ is a substitution of constant symbols $\bar{a}$, $[\bar{x} \mapsto \bar{a}]$, such that $\mathcal{K} \models Q[\bar{x} \mapsto \bar{a}]$. □

Proposition 6 (*partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ Query Answering [4])
Query answering over *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ knowledge bases is complete for PTIME (data complexity).

The query answering algorithm presented in [4] requires both ABox completion and query reformulation. The former is needed to propagate concept memberships along feature chains present in the data when implied by a TBox, and the latter is needed to avoid the need for potentially exponentially many witnesses of anonymous objects. Note that the ABox completion also deals with *equalities* stipulated in the ABox and/or generated by PFD-based subsumptions in the knowledge base TBox.

3 Referring Expressions and CBoxes

In this section we introduce *referring expressions*, concept descriptions that will serve as external identifiers of objects in *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ knowledge bases.

We will require that these concept descriptions *behave* the same way *constant symbols* behave in the traditional setting: we expect their interpretations to be *singular* in every model of a given knowledge base.

Definition 7 (Referring Expressions and Singularity)
Let C be a *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ concept description conforming to the grammar

$$C ::= A \mid C_1 \sqcap C_2 \mid \exists f.C \mid \exists f^{-1}.C \mid \{a\},$$

where a is a constant symbol, and $\mathcal{T}$ a TBox. We say that C is a *(singular) referring expression (w.r.t. $\mathcal{T}$)* if $|C^{\mathcal{I}}| \leq 1$ for all interpretations $\mathcal{I}$ that are models of $\mathcal{T}$. □

We use referring expressions to define the counterpart of *assertions* in traditional knowledge bases.

Example 8
Consider a situation in which objects are identified by their f value, such as an *employee number*, within a class A. Two A objects can then be captured by the following pair of concepts:

$$A \sqcap \exists f.\{123\} \quad \text{and} \quad A \sqcap \exists f.\{345\}.$$

The *singularity requirement*, in this example, can be enforced by ensuring f-values can indeed serve as the key for A objects, e.g., by adding the inclusion dependency

$$A \sqsubseteq A : f \to id$$

to the TBox. The two concepts now replace the usual ABox assertions of the form $A(c_1)$ and $A(c_2)$, which need an invention of additional constant symbols for the two abstract objects. Moreover, ABox assertions of the form $g(c_1) = c_2$ can also be captured by concepts, in our example:

$$A \sqcap \exists f.\{123\} \sqcap \exists g.(A \sqcap \exists f.\{345\}).$$

Note that such descriptions naturally arise when the assertion part of a knowledge base is captured in various database back-ends, e.g., in relational databases (via keys and foreign keys) or in document databases, such as MongoDB[2], in which the structure of the referring expressions correspond to JSON[3].

Example 9
To illustrate the flexibility of our approach based on CBoxes, consider the following JSON fragment describing persons, hypothetically occurring in a MongoDB document source:

```
{"fname" : "John", "lname" : "Smith", "age" : 25,
  "phoneNum" : [
     {"loc" : "home", "dialnum" : "212 555-1234"},
     {"loc" : "work", "dialnum" : "212 555-4567"}
   ]}
```

In our setting, this document can be naturally and directly represented as a CBox assertion of the form

$$\begin{aligned}\text{PERSON} \sqcap (\exists \mathit{fname}.\{\text{“John”}\}) \sqcap (\exists \mathit{lname}.\{\text{“Smith”}\}) \sqcap \exists \mathit{age}.\{25\} \\ \sqcap \exists \mathit{phoneNumFor}^{-1}.((\exists \mathit{loc}.\{\text{“home”}\}) \sqcap (\exists \mathit{dialnum}.\{\text{“212 555-1234”}\})) \\ \sqcap \exists \mathit{phoneNumFor}^{-1}.((\exists \mathit{loc}.\{\text{“work”}\}) \sqcap (\exists \mathit{dialnum}.\{\text{“212 555-4567”}\}))\end{aligned}$$

One way in which this assertion satisfies an *admissibility* condition, defined below, happens whenever the combination of an *fname* and an *lname* can serve to *identify* a PERSON.

[2] `https://www.mongodb.com/`.
[3] `https://www.json.org/`.

Identities of documents (and sub-documents) are now captured using referring expressions; this potentially allows for join operations on document databases (not typically supported by such systems). Queries navigating JSON documents can be now expressed as conjunctive queries over the *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ representation.

This development leads to a revision of the definition of *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases in which the traditional notion of an ABox is replaced by a CBox: a set of concept descriptions that are referring expressions for individuals the knowledge base *knows* about.

Definition 10 (CBoxes, Knowledge Bases, and Query Answers)
Let $\mathcal{T}$ be a *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ TBox and Q a conjunctive query with answer variables $x_1, \ldots, x_k$. We define a *CBox* $\mathcal{C}$ to be a set of *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ concept descriptions.

A *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ *knowledge base* $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{C})$.

We say that the CBox $\mathcal{C}$ is admissible for $\mathcal{T}$ if $|C^{\mathcal{I}}| \leq 1$ for all $C \in \mathcal{C}$ and all models $\mathcal{I}$ of $\mathcal{T}$.

We say that $\mathcal{K}$ is *consistent* if there is an interpretation $\mathcal{I}$ such that

1. $\mathcal{I} \models \mathcal{T}$, and
2. $|C^{\mathcal{I}}| = 1$ for every $C \in \mathcal{C}$.

We say that $(C_1, \ldots, C_k)$ is a certain answer to Q in $\mathcal{K}$ if

$$\mathcal{K} \models Q \wedge C_1(x_1) \wedge \ldots \wedge C_k(x_k)$$

for $\{C_1, \ldots, C_k\} \subseteq \mathcal{C}$. □

3.1 Identity Resolution

CBoxes inherently represent information about how objects in a knowledge base are identified. Note there is no restriction on how such identification must be captured. In particular, there are no *uniformity conditions* on identification of objects that must hold, such as requiring each object to have a single global identifier in all assertions in the knowledge base.

However, CBoxes allow one to *capture* various resolutions of the heterogeneity of identification, e.g., through *translation tables* or *cross-links* [2]. These can be captured using TBox/CBox assertions as follows:

Example 11
Consider the TBox

$$\begin{array}{l}\{\ A \sqsubseteq B,\ C \sqsubseteq B,\\ \quad A \sqsubseteq A : f \to id,\ B \sqsubseteq B : f, g \to id,\ C \sqsubseteq C : g \to id\\ \quad A \sqsubseteq B : f \to id, C \sqsubseteq B : g \to id \qquad\qquad \},\end{array}$$

and the CBox

$$\{\ \ A \sqcap \exists f.\{3\},\quad B \sqcap \exists f.\{3\} \sqcap \exists g.\{5\},\quad C \sqcap \exists g.\{5\}\ \ \}.$$

Note that the referring expression "$A \sqcap \exists f.\{3\}$" identifies *the same object* as "$B \sqcap \exists f.\{3\} \sqcap \exists g.\{5\}$", due to the second-last TBox subsumption, and in turn as "$C \sqcap \exists g.\{5\}$", due to the last TBox assertion. Thus, the object described by "$A \sqcap \exists f.\{3\}$" should be a certain answer to a conjunctive query $\{x \mid A(x) \wedge C(x)\}$. The same happens for all pairs of referring expressions in the CBox subsumed by A and C, respectively, for which there is a B *cross-link*.

3.2 On Minimal Referring Expressions

In relational databases the notion of *candidate key*, a key that has a *minimal* set of attributes of a relation, is typically used as an external identifier of objects stored in the database.

Our development of a *referring expression* strictly generalizes the notion of a *superkey* in the relational setting: sets of attributes, not necessarily minimal, that identifies an object or entity. We now present a procedure that (syntactically) minimizes a referring expression to obtain minimal co-referring referring expressions that are counterparts to relational candidate keys.

Theorem 12 (Minimal Referring Expressions)
Let $\mathcal{T}$ be a *partial*$-\mathcal{CFDI}_{nc}^{\forall-}$ TBox and C a referring expression w.r.t. $\mathcal{T}$. We say that subconcepts of C of the form A, $\{a\}$, $\exists f.\top$, $\exists f^{-1}.\top$, and $\top \sqcap \top$ are *leaves* of C and write $C[L \mapsto \top]$ for a description C in which a leaf L was replaced by $\top$. Assuming "first-leaf" and "next-leaf" denote functions that successively enumerate all leaves of C, the procedure

1. $L :=$ first-leaf(C);
2. **while** $C[L \mapsto \top]$ is singular w.r.t. $\mathcal{T}$ **do**
3. $\quad C := C[L \mapsto \top]$; $L :=$ next-leaf(C);
4. **done**
5. **return** C;

computes a syntactically-minimal co-referring expression for C. (Note that replacing a leaf by $\top$ may create additional leaves.)

Proof (sketch): Since the $C[L \mapsto \top]$ operation weakens the concept description C, it preserves satisfiability. The algorithm tests for singularity at every step. Hence the result is a minimal referring expression equivalent to C since no additional leaves can be removed. □

The algorithm finds a minimal referring expression in time linear in $|C|$. Analogous to the relational setting, backtracking this algorithm facilitates the discovery of alternative minimal referring expressions, and, also analogous to the relational setting, there can be exponentially many of these.

3.3 Reasoning with CBoxes

Our technique crucially depends on mapping CBoxes to (standard) ABoxes as follows. We begin by defining how individual concepts corresponding to referring expressions are transformed:

$$\begin{array}{rcl}
\textsc{ToABox}(a : C_1 \sqcap C_2) & \mapsto & \textsc{ToABox}(a : C_1) \cup \textsc{ToABox}(a : C_2) \\
\textsc{ToABox}(a : \exists f.C) & \mapsto & \{f(a) = b\} \cup \textsc{ToABox}(b : C), b \text{ fresh} \\
\textsc{ToABox}(a : \exists f^{-1}.C) & \mapsto & \{f(b) = a\} \cup \textsc{ToABox}(b : C), b \text{ fresh} \\
\textsc{ToABox}(a : \{b\}) & \mapsto & \{a = b\} \\
\textsc{ToABox}(a : \mathrm{A}) & \mapsto & \{\mathrm{A}(a)\}, \mathrm{A} \text{ primitive}
\end{array}$$

The ToABox function converts a CBox assertion C to a set of ABox assertions by introducing constant names for all necessary individuals, in particular a constant a_C for the (witness of satisfiability of) C itself. The mapping is then lifted to CBoxes by applying it on all referring expressions in the CBox as follows:

$$\textsc{ToABox}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \textsc{ToABox}(a_C : C) \cup \{a_i \neq a_j \mid a_i, a_j \text{ individuals in } \mathcal{C}, i \neq j\}$$

Note that we make nominals distinct since they correspond to values from a relational backend that typically assumes UNA. Also, one could reuse the textual representation of the concepts to serve as the invented constant names.

Theorem 13 (CBox Admissibility)
Let $\mathcal{T}$ be a *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ TBox and C a concept description. Then C is a singular referring expression w.r.t. $\mathcal{T}$ if and only if the knowledge base

$$(\mathcal{T} \cup \{A \sqsubseteq \neg B\}, \textsc{ToABox}(a : C) \cup \textsc{ToABox}(b : C) \cup \{a : A, b : B\})$$

is inconsistent, where A and B are primitive concepts not occurring in $\mathcal{T}$ and C and a and b are distinct constant symbols.

Proof (sketch): The ToABox mapping expands complex concepts in a CBox to sets of assertions in a corresponding ABox. By case analysis we can show that a model of $(\mathcal{T} \cup \{A \sqsubseteq \neg B\}, \textsc{ToABox}(a : C) \cup \textsc{ToABox}(b : C) \cup \{a : A, b : B\})$ provides an counterexample to C's singularity (w.r.t. $\mathcal{T}$). Moreover, whenever C is not singular, such a model can be constructed by appropriately naming additional individuals in a counterexample to C's singularity. □

It is easy to verify that the CBox in Example 11 is admissible w.r.t. the given TBox since 'f', 'f, g' and 'g' are keys on A, B, and C, respectively.

Theorem 14 (Satisfiability of KBs with CBoxes)
Let $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ be a knowledge base with an admissible CBox $\mathcal{C}$. Then $\mathcal{K}$ is consistent if $(\mathcal{T}, \textsc{ToABox}(\mathcal{C}))$ is consistent.

Proof (sketch): Similar to the argument in the proof sketch in Theorem 13. □

3.4 Query Answering over CBoxes

We now show how query answering over CBox-based knowledge bases can be reduced to the standard case of ABoxes. We also show an example of the utility of CBoxes in capturing distinct co-references to a particular object, and how *partial*$-\mathcal{CFDI}_{nc}^{\forall -}$ based TBoxes can account for such co-references.

Theorem 15 (Query Answering)
Let $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ be a consistent knowledge base and $Q = \{(x_1, \ldots, x_k) : \varphi\}$ a conjunctive query over $\mathcal{K}$. Then $(C_1, \ldots, C_k)$ is a certain answer to Q in $\mathcal{K}$ if and only if $(a_{C_1}, \ldots, a_{C_k})$ is a certain answer to Q over $(\mathcal{T}, \textsc{ToABox}(\mathcal{C}))$.

Proof (sketch): Since $C_1, \ldots, C_k$ are singular referring expressions, there must be individuals $o_1, \ldots, o_k$ witnessing nonemptiness of $C_1, \ldots, C_k$, respectively, that make the query true in every model of $\mathcal{K}$; case analysis shows that, in the corresponding models of $(\mathcal{T}, \textsc{ToABox}(\mathcal{C}))$, these individuals will be the interpretations of the constant symbols $(a_{C_1}, \ldots, a_{C_k})$ (and vice versa). □

Note that ABox completion [4, 6] will make constant symbols belonging to *co-referring* referring expressions equal automatically. This, in turn, realizes all reasoning needed to capture the effects of *translation tables* in a TBox/CBox:

Example 16
Reconsider the TBox and CBox in Example 11. The ABox constructed from the CBox is as follows:

$$\begin{array}{l}\{\ A(a_{A\sqcap\exists f.\{3\}}), f(a_{A\sqcap\exists f.\{3\}}) = 3,\\ \quad B(a_{B\sqcap\exists f.\{3\}\sqcap\exists g.\{5\}}), f(a_{A\sqcap\exists f.\{3\}\sqcap\exists g.\{5\}}) = 3, g(a_{A\sqcap\exists f.\{3\}\sqcap\exists g.\{5\}}) = 5,\\ \quad C(a_{C\sqcap\exists g.\{5\}}), f(a_{C\sqcap\exists g.\{5\}}) = 5, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}\end{array}$$

Note that the referring expressions $A \sqcap \exists f.\{3\}$ and $C \sqcap \exists g.\{5\}$, despite being syntactically distinct, *co-refer* to the same object in the knowledge base due to the existence of the B referring expression and the TBox subsumptions. The standard ABox completion [4, 6] then generates the equality

$$\{a_{A\sqcap\exists f.\{3\}} = a_{C\sqcap\exists g.\{5\}}\}$$

using the last two constraints in the TBox and the fact that both A and C are subsumed by B, that serves as a translation concept generated from a translation table. This yields the referring expression $A \sqcap \exists f.\{3\}$ to be a certain answer to the query.

On Query Answers. The definition of *certain answers* asks for all tuples of constants—in our setting proxied by referring expressions—for which the query is entailed by the knowledge base. Thus the selection of referring expressions in the CBox determines components of query answers presented to the user. There are two considerations:

1. Additional answers may be needed; these can be obtained by considering additional referring expressions describing, e.g., sub-documents, to the CBox (as long as the CBox remains admissible);
2. Simpler answers may be desired, i.e., simpler referring expressions denoting the answers; these can be obtained by appropriate selection of minimal referring expressions (and removing all the more complex referring expressions from answers).

Both of these goals can be achieved by a simple housekeeping that determines which referring expressions are eligible to appear in query answers. This step can be easily combined with the CBox-to-ABox mapping by appropriately marking the generated constant symbols. Indeed, similar marking is commonly used, e.g., when ABoxes are *normalized* in most OBDA settings.

4 Summary

We have considered how referring expressions corresponding to concepts in a description logic can serve the role of constant symbols in both assertion boxes and in query answering, and how doing so leads to a more effective and direct way of achieving an integration of backend data sources via OBDA, as well as more descriptive and meaningful answers to queries.

Admitting referring expressions leads naturally to a notion of a concept box or CBox in place of an ABox in a knowledge base. This in turn raises a number of technical issues: how to ensure referring expressions in a CBox refer to a single individual, how to check for knowledge base consistency, and how to evaluate conjunctive queries over the knowledge base. We have shown how all these issues can be resolved by a mapping of CBoxes to ABoxes. The mapping enables off-the-shelf procedures for ABox completion, for consistency checking, and for ABox completion and query rewriting over standard knowledge bases consisting of a TBox and ABox.

In [1], the notion of a referring expression type was also introduced. For future work, we plan to explore how such a typing discipline can be used to push parts of the mapping of CBoxes to ABoxes to backend database sources along the lines outline in [3]. Our new ability of detecting co-reference to objects by referring expressions can also lead to an ability to detect duplicate answers in query results. Future work along this line can enable additional capabilities in query formulation and answering, such as an ability for "limit k" operators in queries.

References

1. Borgida, A., Toman, D., Weddell, G.: On referring expressions in query answering over first order knowledge bases. In: Proc. of KR'16. pp. 319–328 (2016)
2. Calvanese, D., Giese, M., Hovland, D., Rezk, M.: Ontology-based integration of cross-linked datasets. In: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I. pp. 199–216 (2015)

3. Jacques, J.S., Toman, D., Weddell, G.E.: Object-relational queries over $\mathcal{CFDI}_{nc}$ knowledge bases: OBDA for the SQL-Literate. In: Proc. International Joint Conference on Artificial Intelligence, IJCAI. pp. 1258–1264 (2016)
4. McIntyre, S., Borgida, A., Toman, D., Weddell, G.: On limited conjunctions in polynomial feature logics, with applications in OBDA. In: Proc. KR18 (extended abstract). p. (in press) (2018)
5. Toman, D., Weddell, G.E.: On keys and functional dependencies as first-class citizens in description logics. J. Aut. Reasoning 40(2-3), 117–132 (2008)
6. Toman, D., Weddell, G.E.: On partial features in the *DLF* family of description logics. In: PRICAI 2016: Trends in Artificial Intelligence - 14th Pacific Rim International Conference on Artificial Intelligence, Phuket, Thailand, August 22-26, 2016, Proceedings. pp. 529–542 (2016)