Accessing Document Data Sources using Referring Expression Types

Alexander Borgida[1], Enrico Franconi[2], David Toman[3] and Grant Weddell[3]

[1]Department of Computer Science, Rutgers University, New Brunswick, US
[2]KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Italy
[3]Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

Abstract
We show how JSON documents can be abstracted as concept descriptions in an appropriate description logic. This representation allows the use of additional background knowledge in the form of a TBox and an assignment of referring expression types (RETs) to certain primitive concepts to detect situations in which subdocuments, perhaps multiple subdocuments located in various parts of the original documents, capture information about a particular conceptual entity. Detecting such situations allows for normalizing the JSON document into several separate documents that capture all information about such conceptual entities in separate documents. This transformation preserves all the original information present in the input documents. The RET assignment contributes a set of possible concept descriptions that enable more refined and normalized capture of documents, and to more crafted answers to queries that adhere to user expectations expressed as RETs. We also show how RETs allow checking for a document admissibility condition ensuring that each document describes a single conceptual entity.

1. Introduction and Motivation

Suppose we have a JSON/mongoDB document, and an ontology attaching semantics to (most) fields in JSON objects (called “keys” in the JSON definition; for the rest of this paper we reserve the word “key” for database-like keys). More concretely we treat the fields in JSON objects as (functional) roles in an underlying DL and use a TBox to add additional appropriate concept identifiers for the domain of the document. For example, for JSON document

```
{ "fname": "John", "lname": "Smith", "age": 25,
  "wife": { "fname" : "Mary", "lname": "Smith" } }
```

the TBox might contain subsumptions stating, e.g., that:

- PERSONs are objects that posses *fname* and *lname* fields,
- *fname* and *lname* form a key for PERSONs,
- a *wife* of a PERSON is also a PERSON, and so on.

A formal way of capturing these constraints in our FunDL dialect is given in Example 3 below.

DL 2022: 35th International Workshop on Description Logics, August 7–10, 2022, Haifa, Israel
borgida@cs.rutgers.edu (A. Borgida); franconi@inf.unibz.it (E. Franconi); david@uwaterloo.ca (D. Toman); gweddell@uwaterloo.ca (G. Weddell)

We are interested in asking conjunctive queries over the document, with answers being returned in some "answer language" $\mathcal{L}_{answer}$. We could simply create distinct identifiers for each node in the JSON tree, and use roles (obtained from the fields of JSON objects) to connect them, resulting in an ABox, which in the above case might contain an assertion `(id434,25) : age`. In this case, a simple query like `q(P) :- age(P,25)` would return `id434`, which is quite meaningless without laboriously tracing back the creation of the ABox; and this answer makes no connection with the knowledge in the ontology. A more desirable answer would use the TBox, and return something like *PERSON with fname="John" and lname="Smith"*. This issue, and many others concerning the appropriate choice of description for objects in query answers was first addressed in [1], where the notion of "singular referring expression" was introduced. It stood for a *concept description* that was guaranteed to denote a single individual.

Given the use of a referring expression in $\mathcal{L}_{answer}$ to, among others, avoid meaningless object ids, another paper [2] suggested doing a similar thing in the $\mathcal{L}_{tell}$: replacing the ABox with a "CBox", where singular referring expressions are used to state assertional facts as concepts the KB knows about. For example, the JSON fragment above could be captured by the concept

$$\exists\,\texttt{fname}.\{\texttt{"John"}\} \sqcap \exists\,\texttt{lname}.\{\texttt{"Smith"}\} \sqcap \exists\,\texttt{age}.\{\texttt{"25"}\} \sqcap \\ \exists\,\texttt{wife}.(\exists\,\texttt{fname}.\{\texttt{"Mary"}\} \sqcap \exists\,\texttt{lname}.\{\texttt{"Smith"}\})$$

Here, nominal concepts play a central role, as do key constraints for singularity, and a general rule that CBox concepts cannot have empty interpretations.

An algorithm was given in [2] for computing certain answers to conjunctive queries, which returns singular referring expressions appearing in the CBox. This work was carried out in a dialect of the FunDL family of description logics [3]. FunDL and its dialects replace roles with (possibly partial) unary functions called *features*, reify general roles, and always include a so-called *path functional dependency* (PFD) concept constructor to express constraints that are equality generating (including keys). In particular, the dialect used for these purposes in [2] has the property that DL reasoning and query answering are both polynomial-time (in the size of the underlying knowledge base). The above paper also gave, as an example of CBox use, one way to represent a JSON document (in particular, for a document in a MongoDB collection), as a singular concept description in FunDL.

Example 1 (from [2]). *Consider the case where* PERSON *is the name of a MongoDB JSON collection with a value as given in Figure 1. The* PERSON *document is captured by a CBox containing the following concept description expressed in terms of FunDL.*

$$\begin{array}{l}
\exists\,\texttt{collection}.\{\texttt{"person"}\} \sqcap \\
\exists\,\texttt{data}.\exists\,\texttt{dom}^{-}.\exists\,\texttt{ran}(\\
\qquad\quad \exists\,\texttt{fname}.\{\texttt{"John"}\} \sqcap \exists\,\texttt{lname}.\{\texttt{"Smith"}\} \sqcap \\
\qquad\quad \exists\,\texttt{age}.\{\texttt{"25"}\} \sqcap \exists\,\texttt{wife}.\exists\,\texttt{fname}.\{\texttt{"Mary"}\} \sqcap \\
\qquad\quad \exists\,\texttt{phone}.\exists\,\texttt{dom}^{-}.\exists\,\texttt{ran}(\exists\,\texttt{colour}.\{\texttt{"red"}\} \sqcap \exists\,\texttt{dnum}.\{\texttt{"212 555-1234"}\})) \sqcap \\
\qquad \exists\,\texttt{dom}^{-}.\exists\,\texttt{ran}(\\
\qquad\quad \exists\,\texttt{fname}.\{\texttt{"Mary"}\} \sqcap \exists\,\texttt{lname}.\{\texttt{"Jones"}\} \sqcap \\
\qquad\quad \exists\,\texttt{salary}.\{\texttt{"\$150000CAD"}\} \sqcap \exists\,\texttt{spouse}.\exists\,\texttt{fname}.\{\texttt{"John"}\} \sqcap \\
\qquad\quad \exists\,\texttt{phone}.\exists\,\texttt{dom}^{-}.\exists\,\texttt{ran}(\exists\,\texttt{loc}.\{\texttt{"home"}\} \sqcap \exists\,\texttt{dnum}.\{\texttt{"212 555-1234"}\}) \sqcap \\
\qquad\qquad\quad \exists\,\texttt{dom}^{-}.\exists\,\texttt{ran}(\exists\,\texttt{loc}.\{\texttt{"work"}\} \sqcap \exists\,\texttt{dnum}.\{\texttt{"212 666-4567"}\}))
\end{array} \qquad \square$$

```
{ "collection": "person",
 "data" : [
  { "fname": "John", "lname": "Smith", "age": 25,
    "wife": { "fname" : "Mary" },
    "phone": [
      {"colour": "red", "dnum": "212 555-1234"}
     ] } ,
  { "fname": "Mary", "lname": "Jones", "salary": "$150,000 (CAD)",
    "spouse": { "fname": "John" },
    "phone": [
      {"loc": "home", "dnum": "212 555-1234"},
      {"loc": "work", "dnum": "212 666-4567"}
     ] }
 ] }
```

Figure 1: JSON *PERSON* Document.

Intuitively, JSON values are mapped to concepts as follows: (a) primitive values to nominals, (b) (compound) objects to conjunctions of existential restrictions of features corresponding to the field names of the (mapping of) values, and (3) arrays (treated as sets) to multi-valued *roles* reified via the features `dom` and `ran` (see Definition 4).

One can infer that a document would have the above-mentioned singularity property by asserting in a FunDL TBox that collection names (`person` in our case) are unique.

Since in our case query answers are elements of the CBox, the above translation of a full JSON document into a single CBox entry is undesirable. For this reason, we propose to break it up into smaller conceptual entities that make sense based on the terminology in the TBox. This is achieved by using *referring expression types* (RETs) introduced in [1]. In particular, we will show how a combination of a TBox and an RET assignment to the primitive concepts occurring in the TBox enable mapping an initial CBox directly obtained from a MongoDB database, as illustrated above, to an alternative *normalized* CBox, as now illustrated.

Example 2 (JSON normalization)**.** *Applying our normalization procedure to the CBox in Example 1 will then obtain the following concepts:*

$$\begin{aligned}
&\underline{\mathrm{DOCUMENT} \sqcap \exists\, \texttt{collection}.\{\texttt{"person"}\}} \sqcap \\
&\qquad \exists\, \texttt{data}(\exists\, \texttt{dom}^{-}.\exists\, \texttt{ran}(\underline{\mathrm{PERSON} \sqcap \exists\, \texttt{fname}.\{\texttt{"John"}\} \sqcap \exists\, \texttt{lname}.\{\texttt{"Smith"}\}}) \sqcap \\
&\qquad\qquad \exists\, \texttt{dom}^{-}.\exists\, \texttt{ran}(\underline{\mathrm{PERSON} \sqcap \exists\, \texttt{fname}.\{\texttt{"Mary"}\} \sqcap \exists\, \texttt{lname}.\{\texttt{"Jones"}\}}))
\end{aligned}$$

$$\begin{aligned}
&\underline{\mathrm{PERSON} \sqcap \exists\, \texttt{fname}.\{\texttt{"John"}\} \sqcap \exists\, \texttt{lname}.\{\texttt{"Smith"}\}} \sqcap \\
&\qquad\qquad \exists\, \texttt{age}.\{\texttt{"25"}\} \sqcap \exists\, \texttt{wife}.\exists\, \texttt{fname}\{\texttt{"Mary"}\} \sqcap \\
&\qquad\qquad \exists\, \texttt{phone}.\exists\, \texttt{dom}^{-}.\exists\, \texttt{ran}(\underline{\mathrm{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 555-1234"}\}}) \\
&\underline{\mathrm{PERSON} \sqcap \exists\, \texttt{fname}.\{\texttt{"Mary"}\} \sqcap \exists\, \texttt{lname}.\{\texttt{"Jones"}\}} \sqcap \\
&\qquad\qquad \exists\, \texttt{salary}.\{\texttt{"\$150000CAD"}\} \sqcap \exists\, \texttt{spouse}.\exists\, \texttt{fname}\{\texttt{"John"}\} \sqcap \\
&\qquad\qquad \exists\, \texttt{phone}.\exists\, \texttt{dom}^{-}.\exists\, \texttt{ran}(\underline{\mathrm{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 555-1234"}\}}) \sqcap \\
&\qquad\qquad\qquad \exists\, \texttt{dom}^{-}.\exists\, \texttt{ran}(\underline{\mathrm{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 666-4567"}\}}))
\end{aligned}$$

$$\underline{\mathrm{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 555-1234"}\}} \sqcap \exists\, \texttt{loc}.\{\texttt{"home"}\} \sqcap \exists\, \texttt{colour}.\{\texttt{"red"}\}$$
$$\underline{\mathrm{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 555-4567"}\}} \sqcap \exists\, \texttt{loc}.\{\texttt{"work"}\}$$

In the above, the underlined parts of these concepts serve as referring expressions *identifying entities, while the remainder of the concept tells us facts about the entities, using (the dash-underlined) referring expressions when needed, to record such facts. For example, observe how references to phone entities in* phone *facts about persons require only* dnum *facts about phones.* □

Our contributions are as follows.

1. We show how a JSON document (or a MongoDB collection) can be abstracted as a concept description, as illustrated in Example 1, and contrast this way of attaching a meaning or semantics with other proposals.
2. We show how an TBox can attach meaning to such concept descriptions (and therefore to the JSON documents).
3. We describe the normalization procedure which uses the given TBox and a referring expression type assignment in order to extract additional intuitively reasonable CBox subconcepts, as illustrated in Example 2.
4. We also show how information about the same entity can be consolidated even if it was originally recorded in different parts of the JSON document.
5. Finally, we present a more effective way of diagnosing an admissibility property of a CBox that ensures interpretations of referring expressions are indeed singular.

All of the above tasks are accomplished by relying solely on reasoning about equalities and concept memberships of objects corresponding to values in the input JSON document with respect to the DL TBox. This sets our approach apart from many other approaches that commonly rely on mapping and transformation rules.

The paper is organized as follows: Section 2 provides the needed background relating to FunDL and to referring expressions. Section 3 then outlines the main results of this paper relating to the ability to identify subdocuments relating to identifiable entities and subsequent separation of these entities in separate CBox entries/documents. We conclude with a brief overview of related work and with suggestions for follow-on research.

2. Definitions and Background

We now formally define the artifacts introduced in our introductory comments, beginning with a general definition of concept descriptions for members of the FunDL family of DLs with PTIME complexity of logical consequence.[1] Recall that members of this family replace roles with partial functions, and that concept descriptions not only occur in a TBox but also serve as referring expressions in a CBox.

Definition 1 (FunDL Concepts, Referring Expressions and Knowledge Bases)**.** *Let* F *and* PC *be sets of feature names and primitive concept names, respectively. A* path expression *is defined by*

[1]Some additional conditions must hold on PFDs and on conjunctions; see [3, 4] for details.

Syntax	Semantics: Defn of "$(\cdot)^{\mathcal{I}}$"	
$C ::= \bot$	$\emptyset$	(bottom)
$\mid\ C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \to \mathsf{Pf}_0$	$\{x \mid \forall y.((y \in C^{\mathcal{I}} \wedge (\bigwedge_{i=0}^{k} \{x, y\} \subseteq (\exists \mathsf{Pf}_i.\top)^{\mathcal{I}})$ $(\bigwedge_{i=1}^{k} \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y))) \to (\mathsf{Pf}_0^{\mathcal{I}}(x) = \mathsf{Pf}_0^{\mathcal{I}}(y)))\}$	(PFD)
$\mid\ \top$	$\triangle^{\mathcal{I}}$	(top)
$\mid\ \mathrm{A}$	$\mathrm{A}^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$	(primitive concept; $\mathrm{A} \in \mathsf{PC}$)
$\mid\ \exists \mathsf{Pf}.C$	$\{x \mid \exists y.(y \in C^{\mathcal{I}} \wedge \mathsf{Pf}^{\mathcal{I}}(x) = y)\}$	(value restriction)
$\mid\ C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(conjunction)
$\mid\ \{a\}$	$\{a^{\mathcal{I}}\}$	(nominal)
$\mid\ \exists f^{-1}.C$	$\{f^{\mathcal{I}}(x) \mid x \in C^{\mathcal{I}}\}$	(qualified inverse feature)

Figure 2: Syntax and semantics of concept descriptions.

the grammar "$\mathsf{Pf} ::= f.\,\mathsf{Pf} \mid id$*" for* $f \in \mathsf{F}$. *A* concept description *is defined by the grammar on the left-hand-side of Fig. 2.*[2]

An inclusion dependency *is an expression of the form* $C_1 \sqsubseteq C_2$, *where* C_i *is parsed by the first six productions in Fig. 2. A* terminology *(TBox)* $\mathcal{T}$ *consists of a finite set of inclusion dependencies. A* referring expression *is a concept description* C *parsed by the last six productions in Fig. 2. A* concept box *(CBox)* $\mathcal{C}$ *consists of a finite set of referring expressions. A knowledge base* $\mathcal{K}$ *is a TBox/CBox pair* $(\mathcal{T}, \mathcal{C})$.

The semantics *of concept descriptions and path expressions is defined with respect to a structure* $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$, *where* $\triangle^{\mathcal{I}}$ *is a domain of "objects" and* $\cdot^{\mathcal{I}}$ *an interpretation function that fixes the interpretations of primitive concepts* A *to be subsets of* $\triangle^{\mathcal{I}}$ *and primitive features* f *to be partial functions* $f^{\mathcal{I}} : \triangle^{\mathcal{I}} \to \triangle^{\mathcal{I}}$. *The interpretation is extended to 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 *as defined in the centre column of Fig. 2.*

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 a TBox* $\mathcal{T}$ *if it satisfies all inclusion dependencies in* $\mathcal{T}$. $\mathcal{I}$ *is a model of a knowledge base* $\mathcal{K}$ $= (\mathcal{T}, \mathcal{C})$, *written* $\mathcal{I} \models \mathcal{K}$, *if it satisfies* $\mathcal{T}$ *and also that* $|C^{\mathcal{I}}| > 0$ *holds for every* $C \in \mathcal{C}$.

Given a TBox $\mathcal{T}$, *a referring expression* C *is* singular with respect to $\mathcal{T}$ *if* $|C^{\mathcal{I}}| \leq 1$ *for all interpretations* $\mathcal{I}$ *that are models of* $\mathcal{T}$.

The logical implication problem *asks if* $\mathcal{K} \models C_1 \sqsubseteq C_2$ *holds, that is, if* $C_1 \sqsubseteq C_2$ *is satisfied in all models of* $\mathcal{K}$. ☐

Definition 2 (Admissibility and Query Answers). *Let* $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ *be a FunDL knowledge base and* $Q = \{(x_1, \ldots, x_k) \mid \varphi\}$ *a conjunctive query. The CBox* $\mathcal{C}$ *is* admissible *for* $\mathcal{T}$ *if each* $C \in \mathcal{C}$ *is a referring expression that is singular with respect to* $\mathcal{T}$. *(Thus, if* $\mathcal{K}$ *is consistent and* $\mathcal{C}$ *is admissible for* $\mathcal{T}$, *then* $|C^{\mathcal{I}}| = 1$ *for any* $C \in \mathcal{C}$ *and any model* $\mathcal{I}$ *for which* $\mathcal{I} \models \mathcal{K}$.)

[2] A variety of equality generating dependencies, including keys, can be expressed with the use of a *path functional dependency* (PFD) concept description generated by the second production of this grammar.

A k-tuple of referring expressions $(C_1, \ldots, C_k)$ *is a certain answer to* Q *in* $(\mathcal{T}, \mathcal{C})$ *if*

$$\mathcal{K} \models \exists x_1, \ldots, x_k.(\varphi \wedge C_1(x_1) \wedge \ldots \wedge C_k(x_k))$$

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

The second artifact introduced in our introduction relates to so-called *referring expression types* (RETs) introduced in [1]. In this earlier work, such types were attached to the free variables of a conjunctive query, and denoted a space of well-formed formulae ψ with one free variable, over a given FO signature consisting of unary and binary predicates, that were eligible referring expressions for the variable. Such types are essentially patterns of possible ψ, in our case, patterns of possible concept descriptions C, and are now attached to primitive concepts by a user defined *referring expression type assignment* (RTA). They determine a set of possible concept descriptions that are eligible referring expressions for subdocuments. We illustrate this below for our running example, but leave the presentation on how this is accomplished to Section 3.

Definition 3 (Referring Expression Types and Assignments)**.** *A* referring expression type *is defined by the following grammar:*[3]

$$Re ::= \mathrm{A} \mid \exists \mathsf{Pf}.Re \mid Re \sqcap Re \mid \{?\} \mid Re\,;Re$$

A referring expression type assignment *(*RTA*) over a TBox* $\mathcal{T}$ *is partial function mapping primitive concepts* A *occurring in* $\mathcal{T}$ *to a referring expression type* $\mathsf{RTA}(\mathrm{A})$*. We define a language of referring expressions inhabiting Re,* $\mathcal{L}(Re)$*, as follows:*

$$\begin{aligned}
\mathcal{L}(\mathrm{A}) &= \{\mathrm{A}\}\\
\mathcal{L}(\exists \mathsf{Pf}.Re) &= \{\exists \mathsf{Pf}.C \mid C \in \mathcal{L}(Re)\}\\
\mathcal{L}(Re_1 \sqcap Re_2) &= \{C_1 \sqcap C_2 \mid C_1 \in \mathcal{L}(Re_1) \text{ and } C_2 \in \mathcal{L}(Re_2)\}\\
\mathcal{L}(\{?\}) &= \{\{b\} \mid b \text{ is a constant symbol}\}\\
\mathcal{L}(Re_1; Re_2) &= \mathcal{L}(Re_1) \cup \mathcal{L}(Re_2)
\end{aligned}$$
□

Example 3 (CBox normalization)**.** *Let CBox* $\mathcal{C}$ *consist of the single referring expression in Example 1 obtained from the MongoDB collection given earlier, and let TBox* $\mathcal{T}$ *consist of the following inclusion dependencies:*

$$\begin{aligned}
(\exists\, \texttt{collection}.\top) \sqcap (\exists\, \texttt{data}.\top) &\sqsubseteq \mathrm{DOCUMENT}\\
(\exists\, \texttt{fname}.\top) \sqcap (\exists\, \texttt{lname}.\top) &\sqsubseteq \mathrm{PERSON}\\
\exists\, \texttt{dnum}.\top &\sqsubseteq \mathrm{PHONE}\\[4pt]
\mathrm{DOCUMENT} &\sqsubseteq \mathrm{DOCUMENT} : \texttt{collection} \rightarrow id\\
\mathrm{PERSON} &\sqsubseteq \mathrm{PERSON} : \texttt{fname}, \texttt{lname} \rightarrow id\\
\mathrm{PHONE} &\sqsubseteq \mathrm{PHONE} : \texttt{dnum} \rightarrow id\\
\mathrm{PERSON} &\sqsubseteq \exists\, \texttt{wife}.\mathrm{PERSON}
\end{aligned}$$

[3]This is a pattern language obtained by abstracting nominals in referring expressions and by admitting a final production to express *preference* among referring expressions [1].

Our normalization procedure presented in the next section, when given the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ together with the following RTA *assignment, will then replace $\mathcal{C}$ in $\mathcal{K}$ by the CBox in Example 2 of our introduction:*

$$\begin{array}{rcl}
\mathsf{RTA}(\text{DOCUMENT}) & = & \text{DOCUMENT} \sqcap \exists\, \texttt{collection}.\{?\} \\
\mathsf{RTA}(\text{PERSON}) & = & \text{PERSON} \sqcap \exists\, \texttt{lname}.\{?\} \sqcap \exists\, \texttt{fname}.\{?\} \\
\mathsf{RTA}(\text{PHONE}) & = & \text{PHONE} \sqcap \exists\, \texttt{dnum}.\{?\}
\end{array}$$

□

Mapping a JSON value to a referring expression in a CBox is straightforward with FunDL concepts. The following definition of ToConcept provides the details. (Recall that Example 1 in our introduction illustrates an invocation of ToConcept on a JSON collection.) Some observations and reminders: (1) this mapping assumes *any* JSON value, including an array, will map to some element of an underlying domain; (2) the mapping relies entirely on interpreting field names in field-value pairs comprising JSON objects as feature names; and (3) arrays (treated as sets) are mapped to multi-valued roles reified via the features dom and ran.

Definition 4 (ToConcept)**.** *An arbitrary JSON value is mapped to a CBox referring expression as follows:*

$$\begin{aligned}
\textit{ToConcept}(\texttt{"s"}) &\mapsto \{\texttt{"s"}\} \\
\textit{ToConcept}(\texttt{null}) &\mapsto \top \\
\textit{ToConcept}(\{\texttt{"k}_1\texttt{"} : v_1, \ldots, \texttt{"k}_n\texttt{"} : v_n\}) &\mapsto \exists\, \texttt{k}_1.\ \textit{ToConcept}(v_1) \ldots \sqcap \exists\, \texttt{k}_n.\ \textit{ToConcept}(v_n) \\
\textit{ToConcept}([v_1, \ldots, v_m]) &\mapsto \exists\, \texttt{dom}^-.\exists\, \texttt{ran}.\ \textit{ToConcept}(v_1) \sqcap \\
&\qquad \ldots \sqcap \exists\, \texttt{dom}^-.\exists\, \texttt{ran}.\ \textit{ToConcept}(v_m)
\end{aligned}$$

where the first case covers all JSON values that are strings, numerics, and Booleans. □

3. CBox Normalization

Our CBox normalization procedure derives from a pair of normalization rules presented in Section 3.2. To enable references to sub-concepts in a concept description required by our formulation of these rules, we define the mapping ToABox.

Definition 5 (ToABox)**.** *An arbitrary referring expression C is mapped to an ABox and a mapping from constant symbols a_i in this ABox to nodes in the syntactic tree of C as follows:*

$$\begin{array}{rcl}
\textit{ToABox}(a : \{b\}) & \mapsto & \{a = b\} \\
\textit{ToABox}(a : \mathrm{A}) & \mapsto & \{\mathrm{A}(a)\}, \mathrm{A}\ \textit{primitive} \\
\textit{ToABox}(a : \exists f.C) & \mapsto & \{f(a) = b\} \cup \textit{ToABox}(b : C), b\ \textit{fresh} \\
\textit{ToABox}(a : \exists f^{-1}.C) & \mapsto & \{f(b) = a\} \cup \textit{ToABox}(b : C), b\ \textit{fresh} \\
\textit{ToABox}(a : C_1 \sqcap \ldots \sqcap C_n) & \mapsto & \bigcup_{i=1}^{n} \{a = b_i\} \cup \textit{ToABox}(b_i : C_i), b_i\ \textit{fresh}
\end{array}$$

We define $\textit{ToABox}(\mathcal{C}) = \bigcup_{C \in \mathcal{C}} \textit{ToABox}(a : C)$, a fresh for each C. □

The ToABox(C) function converts an input concept C to an ABox by traversing the syntactic structure of the concept, assigning *distinct* constant symbols a to subconcepts of C, and then replacing these subconcepts by atomic ABox assertions.

Definition 6 (Context). *Let C be a concept description. We use the notation $C[a' : C']$ to denote a subconcept C' of the concept C where a' is the constant symbol assigned to C' by* ToAbox$(a : C)$*. For the top-level concept we simply use the context $[a : C]$.*

A *projection function* ToRE that uses a referring expression type Re to generate a referring expression C in $\mathcal{L}(Re)$ and holding for a particular individual a in a knowledge base (or producing an undefined value if no such C exists) is now defined. (Observe the use of an auxiliary recursive function with the same name in the definition that takes a path function Pf as a third argument.)

Definition 7 (Projection on Re). *Let $\mathcal{K}$ be a consistent knowledge base and a a constant. We define a* projection function *ToRE(a, Re) for a referring expression Re as the result of the following recursive definition of ToRE(a, Re, id) on the structure of Re:*

$$\begin{aligned}
\mathit{ToRE}(a, \mathrm{A}, \mathsf{Pf}) &= \mathrm{A} \text{ if } \mathcal{K} \models a : \exists \mathsf{Pf}.\mathrm{A}, \text{ undefined otherwise} \\
\mathit{ToRE}(a, \{?\}, \mathsf{Pf}) &= \{b\} \text{ if } \mathcal{K} \models a : \exists \mathsf{Pf}.\{b\} \text{ for some } b, \text{ undefined otherwise} \\
\mathit{ToRE}(a, \exists \mathsf{Pf}'.Re, \mathsf{Pf}) &= \exists \mathsf{Pf}'.\mathit{ToRE}(a, Re, \mathsf{Pf} \,.\, \mathsf{Pf}') \\
\mathit{ToRE}(a, Re_1 \sqcap Re_2, \mathsf{Pf}) &= \mathit{ToRE}(a, Re_1, \mathsf{Pf}) \sqcap \mathit{ToRE}(a, Re_2, \mathsf{Pf}) \text{ if both defined} \\
\mathit{ToRE}(a, Re_1; Re_2, \mathsf{Pf}) &= \mathit{ToRE}(a, Re_1, \mathsf{Pf}) \text{ if defined, } \mathit{ToRE}(a, Re_2, \mathsf{Pf}) \text{ otherwise} \qquad \square
\end{aligned}$$

The following is a simple consequence of this and our previous definitions. Also, see earlier work in [5, 6] for effective ways of computing the second and fourth base cases.

Lemma 1. *For any constant a, any Re, and any consistent $\mathcal{K}$, ToRE$(a, Re) \in \mathcal{L}(Re)$.*

It will also be useful to have a simplification procedure for referring concepts. The following definition of such a procedure will suffice for illustrative purposes, and clearly preserves concept equivalence.

Definition 8 (Concept Simplification). *We write* Simplify(C) *to denote an exhaustive application of the following to referring expression C:*

1. *If an n-way conjunction contains $\exists f.C_1$ and $\exists f.C_2$, replace both conjuncts by $\exists f.C_1 \sqcap C_2$.*
2. *If an n-way conjunction contains duplicate conjuncts, remove one of the conjuncts.* $\square$

3.1. CBox Admissibility

In this subsection, we show how, given a TBox $\mathcal{T}$, one can statically test for admissibility of any CBox obtained by the normalization rules given in Subsection 3.2 that follows. This is achieved by a mapping to a sequence of logical consequence problems for inclusion dependencies expressing functional dependencies with PFDs that are induced by a given RTA assignment. We begin by defining a normalization of an Re that preserves $\mathcal{L}(Re)$.

Definition 9 (Normalized Types; from [1]). *We write* $\mathsf{Norm}(Re)$ *to refer to an exhaustive application of the following rewrite rules to Re:*

$$\begin{aligned}
Re \sqcap (Re_1; Re_2) &\mapsto Re \sqcap Re_1; Re \sqcap Re_2 \\
(Re_1; Re_2) \sqcap Re &\mapsto Re_1 \sqcap Re_1; Re_2 \sqcap Re \\
\exists \mathsf{Pf}.(Re_1; Re_2) &\mapsto \exists \mathsf{Pf}.Re_1; \exists \mathsf{Pf}.Re_2 \qquad \square
\end{aligned}$$

The following are simple consequences: (1) $\mathcal{L}(Re) = \mathcal{L}(\mathsf{Norm}(Re))$, and (2) all *preference operators* (";") are at the top level. We call the maximal ";"-free parts of $\mathsf{Norm}(Ret)$ *preference-free components*.

To statically test for singularity of referring expressions generated by the ToRE function for a particular referring expression type, we use the following auxiliary definitions:

$$\begin{aligned}
\mathsf{Pfs}(\{?\}) &= \{id\} & \mathsf{Con}(\{?\}) &= \top \\
\mathsf{Pfs}(\mathrm{A}) &= \{\,\} & \mathsf{Con}(\mathrm{A}) &= \mathrm{A} \\
\mathsf{Pfs}(\exists \mathsf{Pf}'.Re) &= \{\mathsf{Pf}' \,.\, \mathsf{Pf} \mid \mathsf{Pf} \in \mathsf{Pfs}(Re)\} & \mathsf{Con}(\exists \mathsf{Pf}'.Re) &= \exists \mathsf{Pf}'.\mathsf{Con}(Re) \\
\mathsf{Pfs}(Re_1 \sqcap Re_1) &= \mathsf{Pfs}(Re_1) \cup \mathsf{Pfs}(Re_2) & \mathsf{Con}(Re_1 \sqcap Re_1) &= \mathsf{Con}(Re_1) \sqcap \mathsf{Con}(Re_2)
\end{aligned}$$

These functions extract a set of paths leading to nominals and a concept from a preference-free referring expression type. Altogether, we are now able to formulate the singularity test following the ideas presented in [1], Theorem 20:

Theorem 1. *Let $\mathcal{T}$ be a TBox and Re a referring expression type. Then all referring expressions in $\mathcal{L}(Re)$ are singular if $\mathcal{T} \models \mathsf{Con}(Re') \sqsubseteq \mathsf{Con}(Re') : \mathsf{Pfs}(Re') \to id$ for every preference-free component Re' of $\mathsf{Norm}(Re)$.*

Our static test of admissibility of any CBox generated by our normalization rules then follows by applying the above to any Re in the range of a programmer supplied RTA.

3.2. CBox Normalization Rules

We now have the necessary machinery to present our two rules for normalizing the CBox of a given knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ and referring expression type assignment RTA. Our first main rule extracts sub-concepts of a given CBox concept C as an additional separate CBox concept.

Definition 10 (Subdocument Extraction). *Assume C is a concept in $\mathcal{C}$ which contains a subconcept C' (i.e., $C[a : C']$) corresponding to a JSON object. Also assume two properties hold: that $(\mathcal{T}, \textsc{ToABox}(\mathcal{C})) \models A(a)$, and that $\mathsf{RTA}(A)$ is defined for primitive concept A. We form a new CBox $\mathcal{C}'$ as follows*

$$\mathcal{C}' := \mathcal{C} - \{C[a : C']\} \cup \{C[a : \textit{ToRE}(a, \mathsf{RTA}(A))], \textit{ToRE}(a, \mathsf{RTA}(A)) \sqcap C'\}$$

if $\textit{ToRE}(a, \mathsf{RTA}(A))$ is defined. If an entity A cannot be properly identified, we report a warning. □

Intuitively, we replace a single monolithic concept C in $\mathcal{C}$ in which a subconcept C' was identified as a representation of an A entity by a modified variant of C in which C' has been *replaced* by its *referring expression.* In addition we create a new CBox concept $\mathrm{ToRE}(a, Re(A)) \sqcap C'$ for this entity.

Note that the choice of A above is non-deterministic, but does not affect the soundness of the extraction. However, this non-determinism can result in different referring expressions used to identify the same subdocument. Although beyond the scope this paper, in [7], we described a technique that allows one to diagnose that a given RTA has anticipated any such ambiguity, that any choice of A must still lead to the same syntactic referring expression.[4] We refer to such an RTA as *identity resolving*.

Also note that, due to the properties of referring expressions (singularity in particular) we have $(\mathcal{T}, \text{ToAbox}(\mathcal{C}')) \models a = b$ for the above-introduced concepts $C[a : \text{ToRE}(a, \mathsf{RTA}(A))]$ and $[b : \text{ToRE}(a, \mathsf{RTA}(A)) \sqcap C']$ in $\mathcal{C}'$. Hence, the models of $(\mathcal{T}, \mathcal{C})$ and $(\mathcal{T}, \mathcal{C}')$ will coincide.

The second of our two rules replaces two CBox referring expressions with a single referring expression when co-reference is implied. Note that, were the given RTA identity resolving, we would always have $D = D'$.

Definition 11 (Equivalent Subdocument Merge). *Let $[a : D \sqcap C]$ and $[b : D' \sqcap C']$ be two concepts in $\mathcal{C}$ such that $(\mathcal{T}, \text{ToAbox}(\mathcal{C})) \models a = b$. We replace $\mathcal{C}$ with*

$$\mathcal{C}' := \mathcal{C} - \{[a : D \sqcap C], [b : D' \sqcap C']\} \cup \{[a : D \sqcap \textit{Simplify}(C \sqcap C')\}. \qquad \square$$

Our main results now follow.

Theorem 2. *Let $\mathcal{C}$ be a CBox in a consistent knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ and $\mathcal{C}'$ a CBox obtained by applying the* Subdocument Extraction *or the* Equivalent Subdocument Merge *rules. Then every model of $\mathcal{K}$ is also a model of $(\mathcal{T}, \mathcal{C}')$ and vice versa.*

Strictly speaking, this is not true for $(\mathcal{T}, \text{ToAbox}(\mathcal{C}))$ and $(\mathcal{T}, \text{ToAbox}(\mathcal{C}'))$, but the models of these two knowledge bases will only differ in what constants are assigned to what subconcepts in $\mathcal{C}$ and $\mathcal{C}'$, respectively.

Theorem 3. *Let $\mathcal{C}$ be an admissible CBox in a consistent knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{C})$ and $\mathcal{C}'$ a CBox obtained by applying the* Subdocument Extraction *or the* Equivalent Subdocument Merge *rules. Then $\mathcal{C}'$ is admissible.*

4. Summary Comments

The main contribution of this paper is showing how JSON-like data sources can be abstracted as concept descriptions in an appropriate DL in a very generic way. This enables their domain-specific semantics to be naturally captured as a TBox in the same logic, which then allows one to draw on mature reasoning services that have been developed for DLs [8, 4]. In addition, our approach utilizes *referring expressions* [1] as the means of identifying entities described in such data sources. This is crucial to the proposal's ability to detect multiple subdocuments that provide information about the same entity. Notably, this is achieved completely automatically since the appropriate equalities will be *entailed* by the knowledge base consisting of a domain-specific TBox and the data sources captured as concepts in a CBox.

Our primary technical contribution is our ability to separate entities into separate documents and to consolidate documents that provide information about the same entity, also by appeal to referring expressions and to entailment in the underlying DL.

[4]Essentially, this entails ensuring that preference in $\mathsf{RTA}(A)$, for any primitive concept A, exhaustively accounts for any primitive concept B for which there exists some interpretation $\mathcal{I}$ for which $(A \sqcap B)^{\mathcal{I}}$ is non-empty.

4.1. Related Work

There are many papers that take as input a semi-structured document, in JSON or XML (sometimes with a schema) plus an ontology in a DL TBox, and create individual instance descriptions in an ABox; for a survey see [9]. Usually, this is intended for just adding semantics to the document, but reasoning could be used to detect inconsistencies, such as a situation where two properties with disjoint domains apply to the same individual. For example, the use of XPath to navigate XML documents and detect instances of objects belonging to certain OWL classes has been presented in [10]. In particular, their approach and implementation, i.e., their JXML2OWL framework, maps XML documents to existing OWL ontologies via explicit mapping rules that use XPath. Out approach, in contrast, uses a generic mapping of JSON to concept descriptions in a DL and then uses the full power of reasoning in the DL to capture such mappings and to achieve a variety other goals, including detection of entities and entity-based equality between subdocuments.

The closest to our work is a virtual OBDA architecture, where data is stored in a JSON MongoDB repository [11]. The architecture extends the basic OBDA architecture by introducing a relational view (an ABox) over MongoDB with respect to a set of type constraints. Rewritten queries by the OBDA framework over the relational view are translated using a fragment of MongoDB aggregate queries. However, as far as we are aware, functional dependencies are not involved in identification issues, nor in separating entities into separate documents.

There is a great deal of work that attempts to synthesize schemata (in various formalisms) from the semi-structured data [9]. However, these approaches are orthogonal to the results in our paper.

4.2. Future work and Extensions

There are many avenues for future research:

1. Additional CBox entries generated by reified roles: in Example 2, we could also have created additional CBox entries for (reified) *phone ownership*, e.g.,

$$\underline{\text{HAS-PHONE} \sqcap \exists\, \texttt{dom.phone}^{-1}.\underline{(\text{PERSON} \sqcap \exists\, \texttt{fname}.\{\texttt{"John"}\} \sqcap \exists\, \texttt{lname}.\{\texttt{"Smith"}\})}}$$
$$\underline{\sqcap \exists\, \texttt{ran}.\underline{(\text{PHONE} \sqcap \exists\, \texttt{dnum}\{\texttt{"212 555-1234"}\})}}.$$

   This, however, requires generalizing the PFD concept constructor and path descriptions to allow for a limited use of inverse features.
2. Diagnosis via consistency and pinpointing/data cleaning: inconsistency of the knowledge base consisting of the domain knowledge $\mathcal{T}$ and the CBox ToConcept(doc) indicates that either our domain knowledge does not accurately capture the properties of the documents or that the documents themselves contain erroneous data. Axiom pinpointing [12] and data cleaning [13] can help in this situation.
3. Set-valued properties and referring expressions: another extension relates to extending the RTAs to allow identification to be based on set-valued properties/values. Such an extension, however, requires extensions to the equality-generating constraints in the underlying DL and the ToRE operation.

References

[1] A. Borgida, D. Toman, G. Weddell, On referring expressions in query answering over first order knowledge bases, in: Proc. KR, 2016, pp. 319–328.

[2] D. Toman, G. E. Weddell, Identity resolution in ontology based data access to structured data sources, in: A. C. Nayak, A. Sharma (Eds.), PRICAI 2019: Trends in Artificial Intelligence - 16th Pacific Rim International Conference on Artificial Intelligence, Part I, volume 11670 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 473–485.

[3] S. McIntyre, D. Toman, G. E. Weddell, FunDL - A family of feature-based description logics, with applications in querying structured data sources, in: Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, 2019, pp. 404–430.

[4] S. McIntyre, A. Borgida, D. Toman, G. E. Weddell, On limited conjunctions and partial features in parameter-tractable feature logics, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, 2019, pp. 2995–3002.

[5] J. Pound, D. Toman, G. E. Weddell, J. Wu, Query algebra and query optimization for concept assertion retrieval, in: V. Haarslev, D. Toman, G. E. Weddell (Eds.), Proceedings of the 23rd International Workshop on Description Logics (DL 2010), Waterloo, Ontario, Canada, May 4-7, 2010, volume 573 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2010.

[6] J. Pound, D. Toman, G. E. Weddell, J. Wu, An assertion retrieval algebra for object queries over knowledge bases, in: T. Walsh (Ed.), IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, IJCAI/AAAI, 2011, pp. 1051–1056.

[7] A. Borgida, D. Toman, G. E. Weddell, On referring expressions in information systems derived from conceptual modelling, in: I. Comyn-Wattiau, K. Tanaka, I. Song, S. Yamamoto, M. Saeki (Eds.), Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings, volume 9974 of *Lecture Notes in Computer Science*, 2016, pp. 183–197.

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.

[9] M. Hacherouf, S. N. Bahloul, C. Cruz, Transforming xml documents to owl ontologies: A survey, Journal of Information Science 41 (2015) 242–259.

[10] T. Rodrigues, P. Rosa, J. Cardoso, Mapping xml to exiting owl ontologies, in: International Conference WWW/Internet, Citeseer, 2006, pp. 72–77.

[11] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, G. Xiao, OBDA beyond relational dbs: A study for mongodb, in: M. Lenzerini, R. Peñaloza (Eds.), Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016, volume 1577 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016.

[12] R. Peñaloza, Axiom pinpointing, in: G. Cota, M. Daquino, G. L. Pozzato (Eds.), Applications and Practices in Ontology Design, Extraction, and Reasoning, volume 49 of *Studies on the Semantic Web*, IOS Press, 2020, pp. 162–177.

[13] I. F. Ilyas, X. Chu, Data Cleaning, ACM, 2019. URL: https://doi.org/10.1145/3310205. doi:10.1145/3310205.