A Formal Framework for Coupling Document Spanners with Ontologies

Domenico Lembo
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Rome, Italy
lembo@diag.uniroma1.it

Federico Maria Scafoglieri
Dip. di Ingegneria Informatica, Automatica e Gestionale
Sapienza Università di Roma
Rome, Italy
scafoglieri@diag.uniroma1.it

***Abstract*—A significant portion of information that is nowadays collected in enterprises and organizations resides in text documents, and thus is inherently unstructured. Turning it into a structured form is the aim of information extraction (IE). Depending on the approach followed, the output of an IE process can fill forms or populate relational tables, or can be presented through an ontology. This last approach is particularly interesting, since ontologies may facilitate the integration with other corporate and external data, and enable data management and governance at an abstract, conceptual level, as in Ontology-based Data Access (OBDA). To this aim, OBDA uses declarative mappings that specify the relation between the ontology and the database to be accessed. In OBDA, however, only mappings towards relational databases have been so far considered, and how to declaratively relate the ontology to unstructured sources is still unexplored. By leveraging the study on *document spanners* for IE, in this paper we propose a new framework that allows to map text documents to ontologies, in the spirit of the OBDA approach. We then investigate the problem of answering conjunctive queries (CQs) in our framework, and show that, if the ontology is specified in the lightweight Description Logic *DL-Lite*$_R$, the problem can be solved by reformulating the user query into a new spanner. Interestingly, both the spanners used in the mapping and the one computed by the rewriting algorithm have the same expressiveness, and CQ answering in this case is polynomial in data complexity.**

I. Introduction

Ontologies have established themselves over the years as one of the best means to represent and share knowledge. At the same time they proved to be an excellent tool for accessing and governing data, when they are properly connected to databases, as in the popular Ontology-based Data Access (OBDA) framework [1], [2]. OBDA is based on a three-level architecture, where the ontology is the highest layer providing a conceptual representation of the business domain, whereas the *mapping* is the intermediate layer, which declarative specifies the relationship between the ontology and the data sources, i.e., the lowest level of the architecture. The users interact only with the ontology, e.g., by formulating their queries, which are automatically processed by sophisticated reasoning algorithms.

In OBDA, however, Ontologies have been essentially used so far only on top of relational databases, with very few exceptions (as, e.g., [3]). In many contexts, however, a substantial portion of relevant business information is nowadays collected in the form of *unstructured data*, and in particular in free-text documents, such as reports, e-mails, legal acts, web pages, PDFs, etc. These documents are obviously thought to be read by humans, but their huge amount and dimension make often their manual processing extremely difficult, as well as it is almost impossible to integrate the information therein with other corporate data.

Automatically extracting relevant pieces of information from text and organizing them into a structured representation to ease their understanding and enable further data processing is the aim of information extraction (IE) [4]. IE is an active field of research since the late '80s [5]. The approaches pursued in this area are basically of two types, i.e., statistical or rule-based. Approaches of the first type usually adopt machine learning techniques to learn the probabilistic models they are based on (e.g., [6], [7]), whereas rule-based approaches encode specific extraction tasks into rules, mostly corresponding to finite-state transducers (e.g., [8], [9], [10])[1].

Recently, Fagin et al. have carried out a foundational study on rule-based IE, and introduced a formal framework based on the notion of *(document) spanner* [13], [14]. In brief, a spanner extracts from an input string $\mathbf{D}$ a relation over its *spans*, which are pairs of indices referring to substrings to be extracted from $\mathbf{D}$. Fagin et al. study possible representations of spanners and analyze how the use of some algebraic operations on the relations extracted from strings influences the spanner expressiveness. In particular, they study spanners defined by regular expressions with capture variables (a.k.a. "*regex formulas*") and *relational algebra*.

In this paper we construct on their results and propose a formal framework for coupling document spanners with ontologies. Our basic idea is to adapt the OBDA framework to enable ontologies to be mapped to documents. We thus introduce the notion of Ontology-based document spanning (OBDS) system, in which, an ontology (more precisely, its intensional component) is linked to text documents through what we call *extraction assertions*, which in OBDS act exactly as mapping assertions in OBDA. Intuitively, an extraction assertion associates a document spanner P to a query q

[1]For additional references and a discussion on the literature on IE we refer the reader to [11] and [12].

specified over the ontology, with the intended meaning that the tuples of substrings corresponding to the spans returned by P must be among the answers to q. In our framework we establish queries over the ontology to be conjunctive queries (CQs), whereas document spanners are defined as regex formulas extended with the relational algebra operators union, projection, join and string selection (the class of such spanners is denoted with $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$). We remark that CQs are the most expressive queries for which query answering over ontologies has been shown to be decidable, and spanners in $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$ are among the most expressive document spanners considered in [13]. Notice also that extraction assertions we define resemble Global-Local-As-View (GLAV) mapping assertions used in data integration and in OBDA, and that their semantics, we have sketched above, corresponds to the classical sound interpretation for mappings [15], [16], [17].

After defining our framework, we investigate the problem of query answering, i.e., how to compute the answers to user queries posed over the ontology. We focus on CQs and analyze the case of *DL-Lite*$_R$ OBDS systems, i.e., when the ontology is specified in the Description Logic (DL) *DL-Lite*$_R$ [18], a popular lightweight ontology language that is at the basis of `OWL 2 QL`, one of the tractable profiles of `OWL 2` [19]. Interestingly, in OBDA, when the ontology is in *DL-Lite*$_R$ and mappings are GLAV, CQ answering is first-order rewritable, i.e., it can be reduced to the evaluation of a first-order (i.e., an SQL) query over the underlying database [20]. We recall that an OBDA mapping associates a query over the database to a query over the ontology, and the above result holds even when the queries over the database in the mapping are arbitrary first-order queries. Interestingly, also the rewriting can be specified in first-order, i.e., it has the same expressiveness of database queries used in the mappings (this has a practical fallout, since its evaluation can be delegated to the underlying database engine). A natural question is now whether a similar behaviour shows up also in *DL-Lite*$_R$ OBDS systems, i.e., whether we can reduce query answering to the execution of a document spanner of the same kind of the spanners used in the mapping. We positively answer this question, by providing an algorithm that rewrites every CQ issued over a *DL-Lite*$_R$ OBDS system (i.e., over its ontology) into a spanner belonging to $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$, and thus with the same expressiveness of spanners used in the mapping. Since evaluating such spanners is polynomial in the size of the input document, we can also conclude that CQ answering in this setting is polynomial in data complexity.

We finally observe that the use of ontologies in IE has been already widely considered in the literature (see [21] for a survey on the topic). However, none of the previous works on this matter has proposed a formal declarative framework, nor did they study the problem of query answering as we do in the present paper. Also, we believe that our framework paves the way for an in-depth investigation of the role of ontologies in IE, and in particular for the understanding of how reasoning over the ontology can help IE.

The rest of the paper is organized as follows: In Section II we give some preliminaries; in Section III we introduce our OBDS framework, in Section IV we provide our query rewriting algorithms for OBDSs systems equipped with *DL-Lite*$_R$ ontologies; we conclude the paper in Section V.

II. Preliminaries

A. Ontologies, Description Logics, and Queries

In the context of computer and information sciences, an *ontology* defines a set of representational primitives with which to model a domain of knowledge or discourse. Ontologies are defined in some formal language, which usually has its root in some kind of logic. Description Logics (DLs) are fragments of first-order logic that can be used to represent the knowledge of a domain of interest in a structured and formally well-understood way in order to reason upon it. DLs are thus well-suited to specify ontologies. DLs model the domain of interest in terms of *objects*, i.e., individuals, *concepts*, that are abstractions for sets of objects, and *roles*, that denote binary relations between objects [22]. DL ontologies are widely used in the context of the Semantic Web, and indeed are at the basis of `OWL 2`, the W3C standard for specifying ontologies on the web [23].

In this work we consider ontologies specified in *DL-Lite*$_R$, which is a member of the class *DL-Lite* of tractable DLs, and is the logical basis of the profile `OWL 2 QL` [19]. *DL-Lite*$_R$ expressions are given by the following syntax:

- Concept Expressions:

$$\begin{array}{lcl} B & ::= & A \mid \exists Q \\ C & ::= & B \mid \neg B \end{array}$$

- Role Expressions:

$$\begin{array}{lcl} Q & ::= & P \mid P^- \\ R & ::= & Q \mid \neg Q \end{array}$$

where A denotes an atomic concept (i.e., a named concept from the ontology signature), B a basic concept (i.e, an atomic role A or an existential restriction on a role $\exists Q$), C a general concept (i.e., a basic concept B or its negation $\neg B$), P an atomic role (i.e., a named role from the ontology signature), Q a basic role (i.e., an atomic role P or its inverse P^-), and R a general role (i.e., a basic role Q or its negation $\neg Q$).

A DL ontology $\mathcal{O}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where:

- $\mathcal{T}$, called TBox, is the *terminological component* of $\mathcal{O}$, which contains statements representing intensional knowledge, and
- $\mathcal{A}$, called ABox, is the *assertional component* of $\mathcal{O}$, which represents extensional knowledge.

A TBox in *DL-Lite*$_R$ is a finite set of assertions in form:

$$\begin{array}{ll} B \sqsubseteq C & (\textit{concept inclusion assertion}) \\ Q \sqsubseteq R & (\textit{role inclusion assertion}) \end{array}$$

An Abox in *DL-Lite*$_R$ is a finite set of membership assertions (i.e., facts) of the form:

$$\begin{array}{rl} A(a) & (\textit{concept membership assertion}) \\ P(a,b) & (\textit{role mebership assertion}) \end{array}$$

where a and b are constants (i.e., individual names). The formal semantic of DL language is given in terms of first-order (FOL) interpretations $\mathcal{I}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each role R a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$, and to each individual name a and object $o \in \Delta^{\mathcal{I}}$. Expressions and assertions in *DL-Lite*$_R$ are interpreted $\mathcal{I}$ as showed in the following table:

Construct	Syntax	Semantics
Atomic Concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic Role	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Concept Negation	$\neg B$	$\Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$
Inverse Role	P^-	$\{(o', o) \| (o, o') \in P^{\mathcal{I}}\}$
Existential Rest.	$\exists Q$	$\{o \mid \exists o'.(o, o') \in Q^{\mathcal{I}}\}$
Role Negation	$\neg Q$	$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus Q^{\mathcal{I}}$
Concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
Role Inclusion	$Q \sqsubseteq R$	$Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
Membership Ass.	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
Membership Ass.	$P(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$

An interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O}$ if it satisfies all assertions in $\mathcal{T}$ and $\mathcal{A}$. We denote with $Mod(\mathcal{O})$ the set of all models of an ontology. We also say that $\mathcal{O}$ infers a sentence ψ if $\psi^{\mathcal{I}}$ evaluates to true in every $\mathcal{I} \in Mod(\mathcal{O})$.

A query is an open formula of function free first-order logic (FOL), i.e., a formula of the form:

$$\{\vec{x} \mid \exists \vec{y}.\phi(\vec{x}, \vec{y})\} \tag{1}$$

where $\exists \vec{y}.\phi(\vec{x}, \vec{y})$ is a FOL formula with free variables $\vec{x}$ and existentially quantified variables $\vec{y}$, possibly containing constants. The number of variables in $\vec{x}$ is the *arity* of the query. Among FOL queries, we in particular consider *conjunctive queries* (CQs), i.e., FOL queries in which $\exists \vec{y}.\phi(\vec{x}, \vec{y})$ is a conjunction of the form $\exists \vec{y}.p_1(\vec{x}_1, \vec{y}_1) \wedge \ldots \wedge p_n(\vec{x}_n, \vec{y}_n)$, where each $p_i(\vec{x}_i, \vec{y}_i)$ is an *atom*, $\vec{x} = \cup_{i=1}^n \vec{x}_i$ and $\vec{y} = \cup_{i=1}^n \vec{y}_i$. When queries are specified over an ontology, each predicate p_i in every atom is either an atomic concept or an atomic role from the ontology signature. An extension of *CQs* are *unions of conjunctive queries*. A *union of conjunctive queries* (UCQ) is a FOL query of the form $\{\vec{x} \mid \exists \vec{y}_1.\phi_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_n.\phi_n(\vec{x}, \vec{y}_n)\}$, such that each $\{\vec{x} \mid \exists \vec{y}_i.\phi_i(\vec{x}, \vec{y}_i)\}$ is a CQ. To simplify notation, throughout the paper we can write UCQ as sets of CQs. Query answering over an ontology amount to computing the so-called *certain answers*, i.e., those answers that hold in all models of the ontology. Formally, given a query q of the form (1) over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, a tuple $\vec{c}$ of constants is a certain answer to q if $\mathcal{O} \models \exists \vec{y}.\phi(\vec{c}, \vec{y})$. In the following, we can write $q(\vec{x})$ to denote a query of the form (1) with free variables $\vec{x}$. Also, given a tuple of constants $\vec{c}$, we can write $q(\vec{c})$ to denote $\exists \vec{y}.\phi(\vec{c}, \vec{y})$.

B. Document Spanners

We recall below the framework of *document spanners* studied by Fagin et al. in [13], [14].

1) Strings and spans: We fix a finite alphabet Σ of symbols, which we assume totally ordered. In particular, in the following examples Σ is composed by the lower and capital letters of English alphabet, the full stop ("."), and the underscore ("_"), which stands for the whitespace.

We denote by Σ^* the set of all finite strings over Σ, and by Σ^+ the set of all finite strings of length at least one over Σ. A *language* over Σ is a subset of Σ^*. A document $\mathbf{D}$ is simply a finite string over Σ, i.e., $\mathbf{D} = \sigma_1 \ldots \sigma_n$ with $n \geq 0$ and $\sigma_i \in \Sigma$ for $i \in \{1, \ldots, n\}$. In other terms, $\mathbf{D} \in \Sigma^*$ The length *n* of $\mathbf{D}$ is denoted by $|\mathbf{D}|$. If $n = 0$, then $|\mathbf{D}| = 0$

A *span* identifies a substring of $\mathbf{D}$ by specifying its bounding indices. Formally a *span* of $\mathbf{D}$ has the form $[i, j\rangle$, where $1 \leq i \leq j \leq n+1$. If $[i, j\rangle$ is a span of $\mathbf{D}$, then $\mathbf{D}_{[i,j\rangle}$ denotes the substring $\sigma_i \ldots \sigma_{j-1}$. Note that $\mathbf{D}_{[i,i\rangle}$ is the empty string, and that $\mathbf{D}_{[1,n+1\rangle}$ is $\mathbf{D}$. Two spans $[i, j\rangle$ and $[i', j'\rangle$ are equal if and only if $i = i'$ and $j = j'$.

We denote by $\mathsf{Spans}(\mathbf{D})$ the set of all the spans of $\mathbf{D}$.

Example 1. Consider the document $\mathbf{D}_{\mathbf{ex}}$ given in Figure 1, and the span $[11, 19\rangle$. It identifies the substring $\mathsf{Einstein}$, i.e., $\mathbf{D}_{[11,19\rangle} = \mathsf{Einstein}$. □

We fix an infinite set SVars of (span) *variables*; spans may be assigned to these variables. The sets Σ^* and SVars are disjoint. For a finite set $V \subseteq \mathsf{SVars}$ of variables and document $\mathbf{D} \in \Sigma^*$, a $(V, \mathbf{D})$-*tuple* is a mapping $\mu : V \rightarrow \mathsf{Spans}(\mathbf{D})$ that assigns a span of $\mathbf{D}$ to each variable in V. A $(V, \mathbf{D})$-*relation* is a set of $(V, \mathbf{D})$-*tuples*. A *document spanner* (or just *spanner* for short) is a function P over a finite set $V \subseteq$ *SVars* of variables, that maps a document $\mathbf{D}$ to a $(V, \mathbf{D})$-*relation*.

In the following, we use $\mathsf{SVars}(P)$ to refer to the set of variables of a spanner P. We may also use $P(v_1, \ldots, v_n)$ to denote a spanner P over variables $V = \{v_1, \ldots, v_n\}$. Furthermore, given a document $\mathbf{D}$, we write $\mathsf{eval}(P, \mathbf{D})$ to denote the $(V, \mathbf{D})$-*relation* that maps $\mathsf{SVars}(P)$ to $\mathbf{D}$.

Example 2. In Figure 2 we provide an example of $(V, \mathbf{D})$-*relation*, for the spanner $[\![\gamma_{tok}]\!]$, such that $\mathsf{SVars}([\![\gamma_{tok}]\!]) = \{x\}$. $(V, \mathbf{D})$-tuples listed in this figure correspond to the words of the document $\mathbf{D}_{\mathbf{ex}}$ in Figure 1. □

C. Spanner Representation

By a *spanner representation system* we refer collectively to any manner of specifying spanners through finite objects. In the next subsections we recall the regex formula (a primitive spanner representation) and the spanner algebra (an extension of regex formula with relational algebra operators).

1) Regex Formulas: As defined in [13], a *variable regex* is an extension of a regular expression with *capture variables*. Its grammar is defined as follows:

$$\gamma := \emptyset \mid \epsilon \mid \sigma \mid (\gamma \vee \gamma) \mid (\gamma \cdot \gamma) \mid \gamma^* \mid x\{\gamma\} \tag{2}$$

P	r	o	f	e	s	s	o	r	_	E	i	n	s	t	e	i	n	_	t	a	u	g	h	t	_	p	h	y	s	i	c	s	.
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

T	h	e	_	P	r	o	f	e	s	s	o	r	_	w	o	n	_	a	_	n	o	b	e	l	.
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60

Figure 1. Document $\mathbf{D_{ex}}$

eval($[\![\gamma_{tok}]\!]$, $\mathbf{D_{ex}}$)	
	x
μ_1	$[1, 10\rangle$
μ_2	$[11, 19\rangle$
μ_3	$[20, 26\rangle$
μ_4	$[27, 34\rangle$
μ_5	$[35, 38\rangle$
μ_6	$[39, 48\rangle$
μ_7	$[49, 52\rangle$
μ_8	$[53, 54\rangle$
μ_9	$[55, 60\rangle$

Figure 2. Spanner $[\![\gamma_{tok}]\!]$ applied to the document in Figure 1

The symbol $\emptyset$ defines the empty set, ϵ is the empty string, and $\sigma \in \Sigma$. The $\vee$, $\cdot$, and * symbols denote disjunction, concatenation, and the Kleene-star operators, respectively. $x\{\gamma\}$ instead indicates that the match obtained through the variable regex γ is mapped (in the form of a span) to the variable $x \in \mathsf{SVars}$. Parenthesis may be used in a variable regex in the usual way to specify precedence between operators. We denote by $\mathsf{SVars}(\gamma)$ the set of variables that occur in γ. We use γ^+ as abbreviations $\gamma\cdot\gamma^*$, and $[\sigma_i - \sigma_j]$ as a shortcut for the disjunction of all characters $\sigma \in \Sigma$ such that $\sigma_i \le \sigma \le \sigma_j$. In this paper we consider only variable regex expressions that *functional*, i.e., such that in a matching over a document each variable is associated with one span. A functional variable regex is also called *regex formula*. We denote by RGX the class of regex formulas. For more detail on the interpretation of regex formulas, and the notion functionality, we refer the reader to [13].

Example 3. Consider the following (simplified) regex formulas system:

- $\gamma_{tok} = (\epsilon \vee (\Sigma^* \cdot \_)) \cdot x\{[a{-}zA{-}Z]^+\} \cdot (. \vee \_)$ i.e., a regex formula assigning to x the words in a document (that is, every non-empty sequence of alphabetic characters preceded by either a white space or an empty string, and followed by either a fullstop or a white space);
- $\gamma_{cap} = (\epsilon \vee (\Sigma^* \cdot \_)) \cdot x\{[A - Z] \cdot \Sigma^*\} \cdot (. \vee \_)$, i.e., a regex formula assigning to x the words in a document that begin with a capital letter;
- $\gamma_{aft\_prof} = (\mathsf{Professor} \cdot \_) \cdot x\{\Sigma^+\} \cdot (. \vee \_)$, i.e., a regex formula assigning to x the words in a document that follow the word Professor (plus a white space). ☐

A regex formula γ is naturally viewed as representing a spanner, and by $[\![\gamma]\!]$ we denote the spanner that is represented by γ.

2) Algebra Over Spanner: The class $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$ denotes the class RGX of primitive spanners extended with the three basic operators of positive relational algebra: union ($\cup$), projection (π), and join ($\bowtie$) plus the string selection operator ($\zeta^=$). When we apply these symbols on one or more spanners, the results will be a new spanner. More precisely, let P, P_1 and P_2 be three spanners, such operators are defined as follows [13]:

- **Union.** The union $P_1 \cup P_2$ is defined when P_1 and P_2 are *union compatible*, that is, $\mathsf{SVars}(P_1) = \mathsf{SVars}(P_2)$. In that case, $\mathsf{SVars}(P_1 \cup P_2) = \mathsf{SVars}(P_1)$ and $\mathsf{eval}(P_1 \cup P_2, \mathbf{D})$ = $\mathsf{eval}(P_1, \mathbf{D}) \cup \mathsf{eval}(P_2, \mathbf{D})$
- **Projection.** if $\theta \subseteq \mathsf{SVars}$, then $\pi_\theta(P)$ is the spanner such that $\mathsf{SVars}(\pi_\theta(P)) = \theta$ and $\mathsf{eval}(\pi_\theta(P, \mathbf{D}))$ is obtained from $\mathsf{eval}(P, \mathbf{D})$ by restricting the domain of each $(V, \mathbf{D})$-*tuple* to θ
- **Join.** The join between spanners is defined as $P_1 \bowtie P_2$. It holds that $\mathsf{SVars}(P_1 \bowtie P_2) = \mathsf{SVars}(P_1) \cup \mathsf{SVars}(P_2)$, and $\mathsf{eval}(P_1 \bowtie P_2, \mathbf{D})$ consists of all $(V, \mathbf{D})$-*tuples* μ that agree with some $\mu_1 \in \mathsf{eval}(P_1, \mathbf{D})$ and $\mu_2 \in \mathsf{eval}(P_2, \mathbf{D})$. μ exists if and only if $\mu_1(x) = \mu_2(x)$ for all $x \in \mathsf{SVars}(P_1) \cap \mathsf{SVars}(P_2)$.
- **String selection.** Let R be a k-ary string relation. The string-selection operator ζ^R is parametrized by k variables $x_1, \ldots, x_k$ in $\mathsf{SVars}(P)$, and may then be written as $\zeta^R_{x_1,\ldots,x_k}$. If P' is $\zeta^R_{x_1,\ldots,x_k}$ P, then the span relation $\mathsf{eval}(P', \mathbf{D})$ is the restriction of $\mathsf{eval}(P, \mathbf{D})$ to those $(V, \mathbf{D})$-*tuples* μ such that $(\mathbf{D}_{\mu(x_1)}, \ldots, \mathbf{D}_{\mu(x_k)}) \in R$.

Let ρ be a regex formula, $[\![\rho]\!]$ the spanner represented by ρ, and $\vec{x} = x_1 \ldots x_n$ the sequence of n distinct variables in $\mathsf{SVars}([\![\rho]\!])$. Let $\vec{y} = y_1 \ldots y_n$ be a sequence of n distinct variables. The unary operator rename applied to $[\![\rho]\!]$ returns a new spanner $[\![\rho']\!]$, written also as $[\![\rho[\vec{x}/\vec{y}]]\!]$, obtained by replacing every occurrence of x_i in $[\![\rho]\!]$, with y_i.

Example 4. Using the regex formula defined in the previous section we can define, using the spanner algebra, the following more expressive and complex $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$-spanner.

- $[\![\rho_{prof}]\!] = [\![\gamma_{cap}]\!] \bowtie [\![\gamma_{aft\_prof}]\!]$, i.e., the spanner represented by a regex formula that assigns to the variable x each word that both begins with a capital letter and follows the string Professor_ . The result of applying $[\![\rho_{prof}]\!]$ to the document $\mathbf{D_{ex}}$ in Figure 1 is shown in Figure 3. The extracted span is $[11, 19\rangle$ corresponding to the substring Einstein. ☐

$\mathsf{eval}([\![\rho_{prof}]\!], \mathbf{D_{ex}})$	
	x
μ_1	$[11, 9\rangle$

Figure 3. Result of spanner $[\![\rho_{prof}]\!]$ applied to the document in Figure 1

III. Framework

In this section we introduce our formal framework. An OBDS System $\mathcal{E}$ is a pair $\langle \mathcal{T}, \mathcal{R} \rangle$, where

- $\mathcal{T}$ is a DL TBox.
- $\mathcal{R}$ is a set of *extraction assertions* of the form

$$P(v_1, \ldots, v_n) \rightsquigarrow \Psi(v_1, \ldots, v_n) \qquad (3)$$

  where
  - $P(v_1, \ldots, v_n)$ (the left-hand side of the assertion) is a $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$-spanner.
  - $\Psi(v_1, \ldots, v_n)$ (the right-hand side of the assertion) is a CQ with free variables $v_1, \ldots, v_n$. Atoms of this CQ are built as usual over $v_1, \ldots, v_n$, and possibly over other existentially quantified variables and/or constants, but may also use terms of the form $\mathbf{f}(v_1, \ldots, v_m)$, where $\mathbf{f}$ is a function symbol. The use of terms of this form allows us to denote individuals "constructed" from the spans in $\mathsf{eval}(P, \mathbf{D})$.

An instance of OBDS system is given in Example 5. By focusing on the mapping, we note the presence of terms of the form $\mathbf{f}(v_1, \ldots, v_m)$ (as $\mathbf{prof}(x_1)$). Such terms are useful in all those cases in which the identifiers of individuals that are instances of the ontology do not appear in the underlying documents, but have to be constructed starting from the strings extracted from them. Note that a similar mechanism is adopted in OBDA, and in the W3C standard R2RML for mapping relational tables to RDF datasets, which adopts templates to construct `IRIs` denoting individuals[2].

The semantic of an OBDS system $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ is defined with respect to a document $\mathbf{D}$. Given one such document, an interpretation $\mathcal{I}$ is a *model for $\mathcal{E}$ with respect to* $\mathbf{D}$ if:

- $\mathcal{I}$ is a model for $\mathcal{T}$, and
- $\Psi(\mathbf{D}_{\mu(v_1)}, \ldots, \mathbf{D}_{\mu(v_n)})$ evaluates to true in $\mathcal{I}$ for each mapping $\mu \in \mathsf{eval}(P, \mathbf{D})$.

We use $Mod(\mathcal{E}, \mathbf{D})$ to denote the set of models of $\mathcal{E}$ with respect to $\mathbf{D}$. The notion of logical implication naturally extends to OBDS systems, i.e., given a sentence ϕ we write that $\langle \mathcal{E}, \mathbf{D} \rangle \models \phi$ if $\phi^{\mathcal{I}}$ for every $\mathcal{I} \in Mod(\mathcal{E}, \mathbf{D})$.

Example 5. Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ be a OBDS system where $\mathcal{T}$ is constituted by the following intensional axioms :

$$\begin{array}{rl} t_1: & Course \sqsubseteq \neg Person \\ t_2: & Professor \sqsubseteq Person \\ t_3: & \exists teaches^- \sqsubseteq Course \end{array}$$

Such a TBox states that a course is not a person (t_1), every professor is a person (t_2) and that only course can be taught (t_3). Let $\mathbf{D_{ex}}$ be the document in Figure 1, and let $[\![\rho_{prof}[x_1/x]]\!]$ and $[\![\gamma_{teaches}]\!]$ be two spanner programs. The former is represented by a regex formula that is obtained by a simple renaming of a regex formula given in Example 4 (the renaming will be useful for the examples of next section), whereas the regex formula representing the latter is given below:

- $\gamma_{teaches} = \_ \cdot x_2\{\Sigma^+\} \cdot (\_ \cdot \mathsf{taught} \cdot \_) \cdot y_2\{\Sigma^+\} \cdot (. \vee \_)$, i.e., a regex assigning to x_2 the words before the word taught, and to y_2 the words after taught.

$\mathcal{R}$ is as follows:

$$\begin{array}{ll} M_1: [\![\rho_{prof}]\!](x_1) & \rightsquigarrow Professor(\mathbf{prof}(x_1)) \\ M_2: [\![\gamma_{teaches}]\!](x_2, y_2) & \rightsquigarrow teaches(\mathbf{prof}(x_2), \mathbf{course}(y_2)) \end{array}$$

Notice that both **prof** and **course** are function symbols of arity 1 used to construct individuals from the string returned by the spanners. For instance, if the spanner in M_1 applied to a document returns the string Einstein, the assertions M_1 constructs the individual $\mathbf{prof}(\mathsf{Einstein})$ as an instance of $Professor$. $\square$

In an OBDS system $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$, computing the *certain answers* to a query q with respect to a document $\mathbf{D}$, denoted by $cert(q, \mathcal{E}, \mathbf{D})$, amounts to finding the query answers to q that hold in all models for $\mathcal{E}$ with respect to $\mathbf{D}$. More formally, given a query q of the form (1), a tuple $\vec{c}$ is a certain answer to q with respect to $\mathbf{D}$ if $\langle \mathcal{E}, \mathbf{D} \rangle \models \exists \vec{y}.\phi(\vec{c}, \vec{y})$.

IV. Query Rewriting in *DL-Lite*$_R$ OBDS systems

In this section we consider *DL-Lite*$_R$ OBDS systems, i.e., systems in which the ontology is expressed in *DL-Lite*$_R$, and devise an algorithm to compute the certain answers to a CQ over one such system with respect to a given document $\mathbf{D}$.

To this aim we resort to rewriting techniques used in the context of OBDA, which we slightly adapt to our setting. First of all, we recall the formal definition of OBDA. An OBDA system $\mathcal{J}$ is a triple $\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, where $\mathcal{T}$ is a DL TBox, $\mathcal{S}$ is the relational schema of the source database, and $\mathcal{M}$ is a mapping between $\mathcal{T}$ and $\mathcal{S}$. The mapping $\mathcal{M}$ is a set of assertions of the form

$$\Phi(x_1, \ldots, x_n) \rightsquigarrow \Psi(x_1, \ldots, x_n)$$

where $\Psi(x_1, \ldots, x_n)$ (the right-hand side of the assertion) is exactly as for an extraction assertion in an OBDS system (cf. assertion (3)), whereas $\Phi(x_1, \ldots, x_n)$ (the left-hand side of the assertion) is a FOL query expressed over the schema $\mathcal{S}$. The semantics of OBDA systems is similar to the semantics of OBDS systems, but is defined with respect to a database instance for $\mathcal{S}$. More precisely, given one such database DB, a model for $\mathcal{J}$ is any interpretation $\mathcal{I}$ that satisfies $\mathcal{T}$ and such that for every tuple $(c_1, \ldots, c_n)$ in the evaluation of the query $\Phi(x_1, \ldots, x_n)$ over DB, $\Psi(c_1, \ldots, c_n)$ evaluates to true in $\mathcal{I}$. It is well-known that CQ answering over an OBDA system in which the TBox is expressed in *DL-Lite*$_R$ is FOL-rewritable, i.e., it can be reduced to the evaluation of a FOL query over the source database [18], [20]. That is, it can be reduced to

[2]https://www.w3.org/TR/r2rml/

the evaluation of a query written in the same language of the query used in the left-hand side of mapping assertions, which are the queries that the DBMS managing the source database is able to process.

In the following, by exploiting the similarities between the OBDA and OBDS frameworks, we show that CQ answering in *DL-Lite*$_R$ OBDS systems $\mathcal{J}$ is reducible to the evaluation of a $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$-spanner over a document $\mathbf{D}$, i.e., a spanner of the same expressiveness of those allowed in the left-hand side of the extraction assertions in $\mathcal{J}$.

To maintain the treatment simple, from now on we consider extraction assertions that do not use terms of the form $\mathbf{f}(x_1, \ldots, x_n)$. Intuitively, this corresponds to the assumption that the identifiers of objects that are instances of the ontologies are extracted directly from the documents over which spanners are executed. Our algorithms however can be adapted to manage the presence of terms of this form by using the same techniques showed in [16]. Thus our results apply straightforwardly to general OBDS extraction assertions.

Below we first consider the case in which extraction assertions do not allow for existential variables in their right-hand side, i.e., they resemble so-called Global-As-View (GAV) mappings adopted in data integration. Then we consider the general case, in which extraction assertions are in fact a form of Global-Local-As-View (GLAV) mapping assertions [15]. In the following, we adopt this standard terminology also for extraction assertions.

A. GAV extraction assertions

As previously said, in a GAV extraction assertion there are no existentially quantified variables in its right-hand side, i.e., it is of the form

$$P(\vec{v}) \rightsquigarrow p_1(\vec{v_1}) \wedge \ldots \wedge p_k(\vec{v_k})$$

where each p_i is either an atomic concept or an atomic role in the ontology (notice that in the former case $\vec{v_i}$ is a single variable, in the latter contains two variables), and $\cup_{i=1}^{k} \vec{v_i} \subseteq \vec{v}$. It is easy to see that the previous extraction assertion is equivalent to the set of assertions[3]

$$\begin{array}{rcl} P(\vec{v}) & \rightsquigarrow & p_1(\vec{v_1}) \\ & \cdots & \\ P(\vec{v}) & \rightsquigarrow & p_k(\vec{v_k}) \end{array}$$

We thus assume in the following that GAV extraction assertions are specified in the previous "splitted" form.

Given a GAV *DL-Lite*$_R$ OBDS system $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ and a query q over $\mathcal{E}$, we rewrite q in two steps, which we call *ontology-based rewriting* and *extraction rules-based rewriting*. The first step is aimed at compiling the TBox into the query. The second is aimed at rewriting the query obtained in the first step (which is still a query expressed over the ontology) according to the assertions in $\mathcal{R}$, so that the final result is a document spanner. For the first step we adopt the algorithm PerfectRef presented [18]. According to this algorithm, the positive inclusion assertions in the TBox are used as rewriting rules, from right to left, to repeatedly rewrite atoms in the query. When an atom is rewritten a new CQ is added to the result, as long as a fix point is reached. The final rewriting is indeed a Union of CQs. For example, given a TBox assertion $B \sqsubseteq A$, and a query $\{x \mid A(x)\}$ the atom $A(x)$ is rewritten into $B(x)$ and the query $\{x \mid B(x)\}$ is added to the result. Notice however that for an atom to be rewritten according to an inclusion assertion in $\mathcal{T}$ its terms must respect some syntactic conditions [18]. Moreover, when atoms in the query unify, PerfectRef performs such unification, which may then trigger some further atom rewriting. An example of execution of PerfectRef is given in Example 6.

The second step is through an *unfolding* method, which, roughly, substitutes, in all possible ways, each atom a in the query returned by PerfectRef with the spanners occurring in the left-hand side of extraction assertions referring to a. To this aim, we use the procedure Unfolding, which takes as input a UCQ Q and a set of extraction rules $\mathcal{R}$. This procedure, for each CQ $q \in Q$, each atom $p_i(\vec{t_i})$ in q (where $\vec{t_i}$ is a tuple of terms, i.e., variables and/or constants), and each extraction rule $P(\vec{v}) \rightsquigarrow p_i(\vec{v}_i)$, computes a unification (if it exists) between $p_i(\vec{t_i})$ and $p_i(\vec{v}_i)$, and substitutes $p_i(\vec{t_i})$ with $P(\vec{v})$, modulo the application of the unifier. Notice that only queries having all atoms that unify with at least one extraction assertions are completely unfolded and returned by Unfolding. Finally, Unfolding expresses projections, selections and joins in the queries according to the $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$ syntax. An example of unfolding is given in Example 6.

The rewriting algorithm for the GAV case is given below.

Algorithm OBDS_Rewriting_GAV$(\mathcal{E}, q)$
Input: OBDS $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$, such that $\mathcal{T}$ is a *DL-Lite*$_R$ TBox and $\mathcal{R}$ is a set of GAV extraction assertions, CQ q
Output: $P \in [\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$
begin
  $Q = \mathsf{PerfectRef}(\mathcal{T}, q)$
  $P = \mathsf{Unfolding}(Q, \mathcal{R})$
  return P
end

We can compute the certain answers to CQs over GAV *DL-Lite*$_R$ OBDS systems by means of the algorithm OBDS_Rewriting_GAV, as established by the following theorem.

Theorem 1. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an OBDS system such that $\mathcal{T}$ is a DL-Lite$_R$ TBox and $\mathcal{R}$ is GAV, let $\mathbf{D}$ be a document, let q be a CQ of arity n over $\mathcal{E}$, and let $P(v_1, \ldots, v_n)$ be the spanner returned by* OBDS_Rewriting_GAV$(\mathcal{E}, q)$*. Then, a tuple of constants $(t_1, \ldots, t_n) \in cert(q, \mathcal{E}, \mathbf{D})$ if and only if there exists $\mu \in \mathsf{eval}(P, \mathbf{D})$ such that $t_i = \mathbf{D}_{\mu(v_i)}$ for each $i \in 1, \ldots, n$. Furthermore $P \in [\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$.*

Proof (sketch). The result follows from the following facts: (i) $\mathsf{PerfectRef}(q, \mathcal{T})$ returns the perfect rewriting of a CQ q

[3]With a little abuse of notation, the variables in the left-hand side contain, but are not necessarily equal to, the variables in the right-hand side of each extraction assertions.

with respect to a *DL-Lite*$_R$ $\mathcal{T}$, i.e., given an ABox $\mathcal{A}$ the certain answers to q over $\langle \mathcal{T}, \mathcal{A} \rangle$ coincide with the evaluation of q over $\mathcal{A}$, seen as a database [18]; (ii) the correctness of the unfolding procedure to rewrite queries over the sources in GAV systems [16], and (iii) the fact that $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$ is closed under union, projection, join and string selection [13]. $\square$

Example 6. Consider the setting of Example 5, and the following query q that asks for the persons who teach a course:

$$q = \{x \mid \exists y.\ Person(x) \land teaches(x, y) \land Course(y)\} \quad (4)$$

the result of $\mathsf{PerfectRef}(\mathcal{T}, q)$ is the UCQ Q' containing the following CQs:

$$\begin{aligned} &q'_1 : \{x \mid \exists y.Person(x) \land teaches(x, y) \land Course(y)\} \\ &q'_2 : \{x \mid \exists y.Professor(x) \land teaches(x, y) \land Course(y)\} \\ &q'_3 : \{x \mid Person(x) \land teaches(x, \_)\} \\ &q'_4 : \{x \mid Professor(x) \land teaches(x, \_)\} \end{aligned}$$

The query q'_2 is generated from q'_1 by the rewriting of $Person(x)$ in $Professor(x)$ according to the inclusion assertion (t_2). The query q'_3 is obtained from q'_1 after rewriting $Course(y)$ into $\exists z.teaches(z, y)$ (according to inclusion (t_3)) and after a unifying $teaches(x, y)$ and $teaches(z, y)$. Similarly for q'_4, which is derived from q'_2.

After the execution of $\mathsf{PerfectRef}$, $\mathsf{Unfolding}(Q', \mathcal{R})$ unfolds the queries in Q' using extraction assertions in $\mathcal{R}$. Note that in our example only q'_4 can be completely unfolded, whereas other queries in Q' do not contribute to the final rewriting since some their atoms cannot be unfolded. For q'_4, atom $Professor(x)$ unifies with the atom $Professor(\mathbf{prof}(x))$ in the extraction assertion M_1 and atom $teaches(x, y)$ unifies with the atom $teaches(\mathbf{prof}(x_2), \mathbf{course}(y_2))$ in the extraction assertion M_2. The unfolding will thus produce the following spanner (modulo the application of the function symbol **prof**):

$$[\![\rho_{unf}]\!] = [\![\pi_z([\![\rho_{prof}[z/x]]\!] \bowtie [\![\gamma_{teaches}[z/x_2]]\!])]\!] \qquad (5)$$

$\mathsf{eval}([\![\rho_{unf}]\!], \mathbf{D_{ex}})$, where $\mathbf{D_{ex}}$ is the document in Figure 1, is the span $[11, 19\rangle$. Then, $cert(q, \mathcal{E}, \mathbf{D})$ are the tuples $\mathbf{prof}(\mathsf{eval}([\![\rho_{unf}]\!], \mathbf{D}))$, that in our case is $\mathbf{prof}(\mathsf{Einstein})$. $\square$

B. General extraction assertions

We now consider the case in which we do not pose any restriction on the extraction assertions, i.e., they are GLAV. We start by noting that an extraction assertion $P(v_i, \ldots, v_n) \rightsquigarrow \Psi(v_1, \ldots, v_n)$ can be always transformed into a pair of assertions of the following form

$$\begin{aligned} P(v_i, \ldots, v_n) &\rightsquigarrow R_{aux}(v_1, \ldots, v_n) \qquad (5) \\ R_{aux}(v_1, \ldots, v_n) &\rightsquigarrow \Psi(v_1, \ldots, v_n) \qquad (6) \end{aligned}$$

where R_{aux} is an auxiliary predicate symbol that explicitly denotes the relation returned by the execution of P over a document $\mathbf{D}$, i.e., R_{aux} denotes the set of tuples of strings $(\mathbf{D}_{\mu(v_1)}, \ldots, \mathbf{D}_{\mu(v_n)})$ for each $\mu \in \mathsf{eval}(P, \mathbf{D})$. It is easy to see that the second assertion above is an OBDA GLAV mapping assertion (in fact a LAV assertion [15]), whereas we can treat the first assertion above as a GAV extraction assertion (even though its right-hand side contains an n-ary predicate). Our rewriting algorithm thus proceeds as follows:
(i) we first rewrite the user query according to the ontology (ontology-based rewriting); this steps can be done through $\mathsf{PerfectRef}$ and returns a UCQs over the ontology;
(ii) Each CQ in the rewriting returned by $\mathsf{PerfectRef}$ is rewritten according to the LAV mapping assertions using predicate R_{aux}; to this aim we can use any of the well-known algorithm to rewrite a CQ with respect to a set of LAV mappings in the relational framework [24]; Specifically, we use the $\mathsf{MiniCon}$ algorithm proposed in [25], which, in our framework, returns a UCQ over the predicate $R_{aux\,i}$;
(iii) at this point it remains to simply unfold the query returned by the $\mathsf{MiniCon}$ algorithm; to this aim we can use the $\mathsf{Unfolding}$ procedure we have seen before.

The overal rewriting algorithm is given below. In the algorithm we denote with $\mathcal{R}'$ extraction assertions of the form (5) and with $\mathcal{L}$ LAV mapping assertions of the form (6).

Algorithm $\mathsf{OBDS\_Rewriting}(\mathcal{E}, q)$
Input: OBDS $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$, such that $\mathcal{T}$ is a *DL-Lite*$_R$ TBox
CQ q
Output: $P \in [\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$
begin
$Q_{\mathcal{T}} = \mathsf{PerfectRef}(\mathcal{T}, q)$
$Q_{\mathcal{L}} = \emptyset$
foreach $q \in \mathcal{Q}_{\mathcal{T}}$
$Q_{\mathcal{L}} = Q_{\mathcal{L}} \cup \{\mathsf{MiniCon}(\mathcal{L}, Q_{\mathcal{T}})\}$
$P = \mathsf{Unfolding}(Q_{\mathcal{L}}, \mathcal{R}')$
return P
end

The following theorem can be proved in a way similar to Theorem 1, considered also that $\mathsf{MiniCon}$ returns a perfect rewriting for a CQ over a relational schema with respect to a LAV mapping [25].

Theorem 2. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an OBDS system such that $\mathcal{T}$ is a DL-Lite$_R$, let $\mathbf{D}$ be a document, let q be a CQ of arity n over $\mathcal{E}$, and let $P(v_1, \ldots, v_n)$ be the spanner returned by* $\mathsf{OBDS\_Rewriting}(\mathcal{E}, q)$. *Then, a tuple of constants $(t_1, \ldots, t_n) \in cert(q, \mathcal{E}, \mathbf{D})$ if and only if there exists $\mu \in \mathsf{eval}(P, \mathbf{D})$ such that $t_i = \mathbf{D}_{\mu(v_i)}$ for each $i \in 1, \ldots, n$. Furthermore $P \in [\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$.*

Since executing a spanner in $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\zeta^=\}}]\!]$ is polynomial in the size of the underlying document [13] (i.e., the length of the string corresponding to the document), the following result follows straightforwardly from Theorem 2.

Corollary 1. Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an OBDS system such that $\mathcal{T}$ is a *DL-Lite*$_R$ TBox, let $\mathbf{D}$ be a document and let q be a CQ over $\mathcal{E}$. Then computing $cert(q, \mathcal{E}, \mathbf{D})$ is polynomial in the size of $\mathbf{D}$.

V. Conclusions

The research in the OBDS framework can be continued in many directions. From the theoretical perspective, it would be interesting to investigate query answering in OBDS systems with more expressive languages for the ontology. First candidates are other tractable logics of the *DL-Lite* family, like $DL\text{-}Lite_{\mathcal{A}}$, which extends $DL\text{-}Lite_R$ with functionalities on roles (in a controlled way). It is known that, differently from $DL\text{-}Lite_R$, query answering for $DL\text{-}Lite_{\mathcal{A}}$ TBoxes and GLAV mappings in OBDA is not FOL-rewritable. This is caused by the interaction of existential variables in right-hand side of mapping assertions and functionalities in the TBox. How this interaction affects query answering in OBDS systems, and whether it is possible to solve query answering by rewriting in spanners of the $[\![\mathrm{RGX}^{\{\cup,\pi,\bowtie,\varsigma^{=}\}}]\!]$ class is left for future study. We also plan to investigate query answering in OBDS systems in which the TBox is specified in other DLs for which standard query answering over ontologies is polynomial in data complexity, e.g, $\mathcal{EL}$ [26], or horn DLs [27], [28].

More in general, we believe that our framework paves the way for a comprehensive study on the use of ontologies in IE, and it can help understanding how reasoning services over the ontology may improve IE. For example, we believe that in our framework it is possible to exploit reasoning to identify anomalies in the specification of extraction rules (e.g., inconsistencies), in the spirit of the work on mapping analysis in OBDA [29]. Finally, an obvious future line of research is to develop software tools for OBDS, in order to verify the realizability of our approach in the practice.

References

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Ontology-based data access and integration," in *Encyclopedia of Database Systems, Second Edition*, 2018.

[2] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev, "Ontology-based data access: A survey," in *Proceedings of the Twentyseventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 2018, pp. 5511–5519.

[3] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao, "OBDA beyond relational DBs: A study for MongoDB," in *Proceedings of the Twentynineth International Workshop on Description Logic (DL 2016)*, 2016.

[4] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*, ser. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[5] R. Grishman and B. Sundheim, "Message understanding conference- 6: A brief history," in *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING'96)*, 1996, pp. 466–471.

[6] R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, (ACL 2011)*, 2011, pp. 541–550.

[7] D. Freitag, "Machine learning for information extraction in informal domains," *Machine Learning*, vol. 39, no. 2/3, pp. 169–202, 2000.

[8] H. Cunningham, "GATE, a general architecture for text engineering," *Computers and the Humanities*, vol. 36, no. 2, pp. 223—-254, 2002.

[9] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan, "SystemT: An algebraic approach to declarative information extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, (ACL 2010)*, 2010, pp. 128–137.

[10] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan, "Declarative information extraction using datalog with embedded extraction predicates," in *Proceedings of the Thirtythird International Conference on Very Large Data Bases (VLDB 2007)*, 2007, pp. 1033–1044.

[11] S. Sarawagi, "Information extraction," *Foundations and Trends in Databases*, vol. 1, no. 3, pp. 261–377, 2008.

[12] B. Kimelfeld, "Database principles in information extraction," in *Proceedings of the Thirtythird ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2014)*, 2014, pp. 156–163.

[13] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren, "Document spanners: A formal approach to information extraction," *Journal of the ACM*, vol. 62, no. 2, p. 12, 2015.

[14] ——, "Declarative cleaning of inconsistencies in information extraction," *ACM Transactions on Database Systems*, vol. 41, no. 1, pp. 6:1–6:44, 2016.

[15] M. Lenzerini, "Data integration: A theoretical perspective." in *Proceedings of the Twentyfirst ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, 2002, pp. 233–246.

[16] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati, "Linking data to ontologies," *Journal on Data Semantics*, vol. X, pp. 133–173, 2008.

[17] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: Semantics and query answering," *Theoretical Computer Science*, vol. 336, no. 1, pp. 89–124, 2005.

[18] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family," *Journal of Automated Reasoning*, vol. 39, no. 3, pp. 385–429, 2007.

[19] B. Motik, A. Fokoue, I. Horrocks, Z. Wu, C. Lutz, and B. Cuenca Grau, "OWL Web Ontology Language profiles," World Wide Web Consortium, W3C Recommendation, Oct. 2009, available at http://www.w3.org/TR/owl-profiles/.

[20] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, and M. Ruzzi, "Using OWL in data integration," in *Semantic Web Information Management - A Model-Based Perspective*, 2009, pp. 397–424.

[21] D. C. Wimalasuriya and D. Dou, "Ontology-based information extraction: An introduction and a survey of current approaches," *Information Sciences*, vol. 36, no. 3, pp. 306–323, 2010.

[22] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi, *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[23] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "Owl 2: The next step for owl," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 4, pp. 309–322, 2008.

[24] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*. Morgan Kaufmann, 2012.

[25] R. Pottinger and A. Y. Halevy, "MiniCon: A scalable algorithm for answering queries using views," *Very Large Database Journal*, vol. 10, no. 2–3, pp. 182–198, 2001.

[26] F. Baader, S. Brandt, and C. Lutz, "Pushing the $\mathcal{EL}$ envelope," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005, pp. 364–369.

[27] U. Hustadt, B. Motik, and U. Sattler, "Data complexity of reasoning in very expressive description logics," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005, pp. 466–471.

[28] M. Ortiz, S. Rudolph, and M. Simkus, "Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$," in *Proceedings of the Twentysecond International Joint Conference on Artificial Intelligence (IJCAI 2011)*. IJCAI/AAAI, 2011, pp. 1039–1044.

[29] D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen, "Mapping analysis in ontology-based data access: Algorithms and complexity," in *Proceedings of the fourteenth International Semantic Web Conference (ISWC 2015)*, 2015, pp. 217–234.