Controlled Aggregate Tree Shaped Questions over Ontologies

Camilo Thorne and Diego Calvanese

KRDB Centre
Free University of Bozen-Bolzano
4, Via della Mostra, 39100 (Italy)
{thorne,calvanese}@inf.unibz.it

Abstract. Controlled languages (CLs) are ambiguity-free subsets of natural languages such as English offering a good trade-off between the formal rigor of ontology and query languages and the intuitive appeal of natural language. They compositionally map (modulo a *compositional translation* $\tau(\cdot)$) into (or *express*) formal query languages and ontology languages. Modulo compositionality, they inherit the computational properties of such ontology/query languages. In the setting of OBDAS, we are interested in capturing *query answering* and measuring computational complexity w.r.t. the data queried (a.k.a. *data complexity*). In this paper we focus in defining a CL capable of expressing a subset SQL *aggregate queries*, and study its data complexity w.r.t. several ontology languages and extensions of the query language.

1 Introduction

Ontology-based data access systems (OBDASs) have been proposed by the semantic web community as a way of integrating and querying information coming from heterogeneous sources [16]. Such systems have two main components, *(i)* an ontology and *(ii)* a collection of (possibly multiple) databases of which the ontology, typically an OWL ontology (or an ER, UML, etc., conceptual model), provides a unified, common view or interface. OWL ontologies are formally underpinned by description logics (DLs), which are expressive enough to capture conceptual models [3]. Formal queries, in general fragments of SQL (or SPARQL) queries, such as (unions) of conjunctive queries, are formulated w.r.t. the ontology and later rewritten and evaluated (under OWA) over the datasources [6].

Controlled English interfaces to such systems (targeting non-expert users) have been proposed [5, 9] as a trade-off between the English utterances covered by the system and its performance (measured in terms of, e.g., precision, recall and accuracy), following the desiderata laid by [2] for natural language interfaces to databases. Controlled languages (CLs) are ambiguity-free fragments of a natural language. This allows their being symbolically (and, moreover, compositionally) translated, without any loss of information, into OWL assertions and/or queries to be sent to the back-end OBDAS [5]. The ACE (Attempto Controlled English) CL, and its fragment ACE-OWL (that maps into OWL), is perhaps the best known in the literature [9]. However, the kind of constraints and queries support by OBDASs affect their scalability. Answering e.g. select-project-join SQL queries over OWL ontologies is (at least) worst-case exponential on the data (**coNP**-hard) [3]. These considerations generalize to the CLs supported by OBDAS, in the sense that, modulo compositional translations, they inherit (i.e., *express*) the computational properties of the ontology and query languages.

This raises the issue of how to extend the expressivity/coverage of the CLs without blowing up the performance of the OBDAS. In this paper we propose to consider the class of SQL *aggregate queries*, viz., select-project-join queries with `GROUP BY` and `HAVING` clauses and aggregation functions such as `COUNT`, `SUM`, `MIN`, `AVG`, etc., in combination with several ontology languages and declarative fragments of English. CLs that translate into aggregate queries have been mostly proposed for plain database settings [11]. The main contributions of this paper are the following:

1. We define an interrogative CL, ATQ-English, that compositionally translates into aggregate tree-shaped queries (ATQs). We also look at how to express several ontology languages (that overlap in expressive power with OWL).
2. We consider bare ATQs and extensions (closed under boolean operations and/or equipped with comparisons) together with those ontology languages and study their data complexity. Aggregates do not increase significatively data complexity but syntactic constructs that go beyond selections, joins and projections, do.

2 Tree Shaped Aggregate Queries over Ontologies

Relational Databases. We assume as given a countably infinite *domain* $\Delta := \Delta_O \cup \Delta_V$ partitioned into a domain Δ_O of object constants and a domain Δ_V of values (containing numbers; in what follows $\mathbb{Q}$ and its subsets). We call *tuple* any finite sequence $\bar{c}$ of domain elements. A *database schema* $\mathbf{R}$ is a finite set of relation names. A *relation name* R is a predicate symbol of arity n (a nonnegative integer). A *database instance* (DB) D of $\mathbf{R}$ is a pair $(\Delta, \cdot^D)$ where Δ is the domain and $\cdot^D$ is an *interpretation function* over $\mathbf{R}$, that is, a function mapping each relation symbol R of arity n in $\mathbf{R}$ to a subset R^D of Δ^n, i.e., to a *relation instance*. Observe that databases are basically FO interpretation structures of $\mathbf{R}$ [1]. The *size* $|D|$ of D is defined as the number of tuples in its relation instances. The set of such tuples is denoted *adom*(D).

Conditions. A *term* t is either a variable (like $x, y, ...$) or a constant (like $c, d, ...$). An *atom* is an expression of the form $R(\bar{t})$, where R is a relation name of arity n and $\bar{t}$ is a sequence of n terms. An atom is *ground* when all its terms are constants. A *condition* Φ is a (possibly empty) conjunction of FO atoms closed under negation and existential quantification. We define $\forall\bar{x}\Phi := \neg\exists\bar{x}\neg\Phi$, $\Phi_1 \vee \Phi_2 := \neg(\neg\Phi_1 \wedge \neg\Phi_2)$ and $\Phi_1 \Rightarrow \Phi_2 := \neg\Phi_1 \vee \Phi_2$. We denote by *Var*$(\Phi)$ the set of variables occurring in condition Φ, and by *FV*(Φ) its free variables. A condition is called a *sentence* (or also, *boolean*) if it contains no free variables. A condition $\Phi(x)$ is said to be *tree shaped* if *(i)* it is an atom $A(x)$, *(ii)* it is a condition $\exists y R(x, y)$, *(iii)* it is a condition $\exists y(R(x, y) \wedge \Phi(y))$, where $\Phi(y)$ is tree shaped, or *(iv)* it is a condition $\Phi_1(x) \wedge \Phi_2(x)$ where $\Phi_1(x)$ and $\Phi_2(x)$ are tree shaped. The free variable x of $\Phi(x)$ is called the *root* of Φ.

Ontologies and OBDASs. An *ontology* $\mathcal{O}$ is formally defined as a set of sentences. Ontologies typically express *constraints* (termed also *axioms* or *assertions*) on the data, viz., they provide the conceptual model of a domain of interest, describing the classes of objects such a domain of interest comprises, their attributes or properties, and their relations. An *ontology language* $\mathcal{L}$ is a class of constraints, obtained by suitably restricting their syntax. i.e., a fragment of FO with a distinct expressive power [16]. Ontologies provide a single, unified, global view on datasources for accessing data in ontology-based systems. An *ontology-based data access system* (OBDAS) is a pair $(\mathcal{O}, D)$ where D is a DB and $\mathcal{O}$ is an ontology.

Aggregate Queries. We consider now the following standard SQL *aggregation functions*, viz., **max**, **min**, **count**, **countd**, **sum** and **avg**. In what follows α will denote an arbitrary aggregation function. Given this, we call an *aggregation term* any expression of the form $\alpha(y)$, where y is called an *aggregation variable*. An *aggregate tree-shaped query* (ATQ) over **R** is a query of the form

$$q(x, \alpha(y)) \leftarrow \Phi \tag{1}$$

where q is the *head* relation x is a *grouping variable*, $\alpha(y)$ is an aggregation term and $\Phi := \Phi_1 \wedge R(x, y) \wedge \Phi_2$ with Φ_1 a condition rooted in y, Φ_2 a condition rooted in x, $R(x, y)$ an atom, $\{x, y\} = FV(\Phi)$ and $y \neq x$.

The *core* $\breve{q}$ of an ATQ $q(x, \alpha(y)) \leftarrow \Phi$ is defined as $\breve{q}(x, y) \leftarrow \Phi$. Queries with conjunctive bodies but no aggregation terms in their heads are known in the literature as *conjunctive queries* (CQs). If the sequence of head variables is empty, a CQ is said to be *boolean* [1]. In general, different constraints on the syntax of head relations and conditions give way to different classes of queries. A *query language* $\mathcal{L}$ is any such class.

Certain Answers and QA. Aggregation functions in SQL are defined over bags $\{\!|\cdot|\!\}$, called *groups*, which are collections of possibly repeated symbolic and numerical values (from Δ), and return a rational number. OBDASs and ontologies, on the other hand, deal with incomplete information, i.e., their DBs D are a partial description of the domain of interest that the ontology "completes" by intuitively characterizing the space of all the DBs D' compatible with D [7]. Naively asking an aggregate query q over each D' may thus yield no meaningful answer: it might return a different group and a different quantity over each D' [7]. To overcome this we *(i)* ask for the tuples satisfying the condition of the query over *all* the D's, *(ii)* group those tuples and *(iii)* return the value of the aggregation function over that group [7]. Finally, the associated query answering problem allows us to study computational properties (i.e., data complexity).

An *assignment* γ over a condition Φ is a function that maps $Var(\Phi)$ to Δ and each constant in Φ to itself. Assignments are extended to complex syntactic objects like atoms and conditions in the usual way. An assignment γ is said to *satisfy* a condition Φ over D, denoted $D, \gamma \models \Phi$, whenever Φ evaluates to true in D under γ, i.e., the standard notion of satisfaction in FO [1]. We denote by $Sat_D(\Phi)$ the set of satisfying assignments of Φ over D. A DB D' is said to extend a DB D, if, for each n-ary relation symbol R, $R^D \subseteq R^{D'}$.

Let $(\mathcal{O}, D)$ be an OBDAS. Consider an ATQ q of the form $q(x, \alpha(y)) \leftarrow \Phi$ with grouping variables x, aggregation variables y and condition Φ. Let c be a tuple. A *certain assignment* is a mapping $\delta \colon \mathit{FV}(\Phi) \to \mathit{adom}(D)$ where, for each D' that is a model of $\mathcal{O}$ and extends D, there exists an assignment $\gamma \in \mathit{Sat}_{D'}(\Phi)$ s.t., for all $x \in \mathit{FV}(\Phi)$, $\delta(x) = \gamma(x)$. We denote by $\mathit{Sat}^{\mathcal{O}}_{D}(\Phi)$ this set. The *certain group* of tuple c is the bag

$$H_c := \{\!|\delta(y) \mid c = \delta(x), \delta \in \mathit{Sat}^{\mathcal{O}}_{D}(\Phi)|\!\} \tag{2}$$

and the set of *certain answers* of q over $(\mathcal{O}, D)$ is the set

$$\mathit{cert}(q, \mathcal{O}, D) := \{(\delta(x), \alpha(H_{\delta(x)})) \mid \delta \in \mathit{Sat}^{\mathcal{O}}_{D}(\Phi)\}. \tag{3}$$

The *query answering* (QA) decision problem for ATQs over OBDASs is the decision problem stated as follows: **Input:** a tuple (c, n), an ATQ q an OBDAS $(\mathcal{O}, D)$. **Question:** does $(c, n) \in \mathit{cert}(q, \mathcal{O}, D)$? We are interested in the complexity of QA measured considering D as its sole input, that is, in the so-called *data complexity* of QA [18].

Example 1. Consider the database schema $\mathbf{R}_s$:=$\{$*takesCourse*,*comesFrom*, *takesPlace*, *Country*, *Student*, *University*$\}$, where s in $\mathbf{R}_s$ stands for ”student”. A database of $\mathbf{R}_s$ is the database D_s

takesCourse	
SName	Course
Luca	TOC
Luca	ADS
James	German

Student
SName
Luca
James

University
PName
Unibz

Course	
CName	CCred
TOC	4
ADS	4
German	0

Country
CoName
Italy
UK

takesPlace	
CName	PName
TOC	Unibz
German	LC

comesFrom	
SName	CoName
Luca	Italy
James	UK

where ”TOC” stands for computational complexity and computability theory, ”ADS” for algorithms and data structures, ”LC” for Language Centre and ”Unibz” for Bolzano University. (for convenience we use attribute names to denote relation positions). The following set $\mathcal{O}_s$ of constraints

$$\begin{array}{cc}
\forall x(\exists y \mathit{takesCourse}(x,y) \Rightarrow \mathit{Student}(x)) & \forall x(\exists y \mathit{takesCourse}(y,x) \Rightarrow \mathit{Course}(x)) \\
\forall x(\exists y \mathit{comesFrom}(x,y) \Rightarrow \mathit{Student}(x)) & \forall x(\exists y \mathit{comesFrom}(y,x) \Rightarrow \mathit{Country}(x)) \\
\forall x(\exists y \mathit{takesPlace}(x,y) \Rightarrow \mathit{Course}(x)) & \forall x(\exists y \mathit{takesPlace}(y,x) \Rightarrow \mathit{Place}(x)) \\
\forall x(\exists y \mathit{hasCredits}(x,y) \Rightarrow \mathit{Student}(x)) & \forall x(\mathit{University}(x) \Rightarrow \mathit{Place}(x))
\end{array}$$

provide a conceptual model of the domain of students. Hence, the pair $(\mathcal{O}_s, D_s)$ constitutes an OBDAS. Consider now the **count** ATQ q_0 over $\mathbf{R}_s$

$$\begin{aligned} q_0(x, \mathbf{count}(y)) \leftarrow \exists z(&\mathit{Student}(y) \wedge \mathit{takesCourse}(y,z) \wedge \\ &\mathit{comesFrom}(y,x) \wedge \mathit{Country}(x)) \end{aligned} \tag{q_0}$$

which we would had written in SQL as

```
SELECT s2.CName, COUNT(s2.SName)
FROM Student s1, comesFrom s2
WHERE EXISTS (SELECT *
   FROM takesCourse s3
   WHERE s1.SName = s2.SName AND s2.SName = s3.SName)
GROUP BY s1.CName
```

The query q_0 asks for the number of courses taken by students of each country. Asking q_0 to $(\mathcal{O}_s, D_s)$ gives $\mathit{cert}(q_0, \mathcal{O}_s, D_s) = \{(\text{Italy}, 1), (\text{UK}, 1)\}$. ∎

3 Expressing Query Answering (QA)

Translating English declarations and questions into ontology constraints and queries symbolically and compositionally can be modelled by English formal semantics *compositional translations* $\tau(\cdot)$, in the spirit of Montague and Clifford [12, 8], which use intermediate expressions called *meaning representations* from higher order logic (HOL), i.e., FO enriched with the λ-application, λ-asbtraction, β-reduction and the types of the simply-typed λ-calculus. The function $\tau(\cdot)$ is recursively defined on English components by enriching (formal) grammars with semantic actions [10]: it exploits the syntax of the CL utterance by λ-applying the siblings of every syntactic constituent and by λ-abstracting the free variables introduced (in the MR) by subordinated clauses [10]. For every language L, we define $\tau(L) := \{\tau(w) \mid w \in L\}$.

Given a formal language L', to *express L' in CL* we define a declarative CL L and a compositional translation $\tau(\cdot)$ s.t. $\tau(L) = L'$. Given an ontology language $\mathcal{L}$ and a query language $\mathcal{Q}$, to *express QA in CL* we express $\mathcal{L}$ and $\mathcal{Q}$.

Expressing Ontologies. CLs are tightly linked to ontologies. Traditionally, they were used for tasks such as ontology authoring. More recently, they have been used for declaring and querying information. The OWL web ontology language[1] is the W3C standard for web-based ontologies and ontology-based systems, and is formally underpinned by description logics (DLs), which are decidable fragments of FO specifically tailored for representing and reasoning about knowledge [3]. In particular, OWL 2 corresponds to the DL $\mathcal{SROIQ}$ (with data types). The CL ACE-OWL [9] expresses OWL in controlled English. OWL however does not scale to data in ontology based systems: inference and query answering are **coNP**-hard in data complexity.

It is of interest, then, to consider CLs for which the data complexity of QA is tractable. One such CL is Lite-English (see [4] for its definition), for which QA is as hard as for DBs (in **LogSpace**). Lite-English expresses the DL *DL-Lite* [4]. A meaningful fragment of OWL closed under boolean operations in the DL $\mathcal{ALCI}$, expressed by the CL DL-English [17]. It is also of interest to consider fragments of everyday English whose expressiveness overlaps with OWL. The Fragments of English (FOE) [14] are obtained incrementally by considering all the (grammatical) utterances one can build using only copula, common nouns and the **Det**s ”some”, ”every” and ”no”, i.e., the syllogistic fragment, and then exteding coverage to further English constructs. See Figure 1.

Expressing ATQs. We express ATQs with the CL ATQ-English. Sets are seen, in formal semantics, as characteristic functions of type $e \to t$. Similarly, bags can be seen as functions of type $e \to \mathbb{N}$. To express SQL aggregation functions we use *aggregate determiners* of type $(e \to \mathbb{N}) \to \mathbb{Q}$). They are applied to **N** constituents which are made to denote bags:

$$\begin{aligned} \tau(\text{the greatest number of}) &:= \lambda P^{\mathbb{Q}\to\mathbb{N}}.\mathbf{max}(P){:}(e \to \mathbb{N}) \to \mathbb{Q}, \\ \tau(\text{the smallest number of}) &:= \lambda P^{\mathbb{Q}\to\mathbb{N}}.\mathbf{min}(P){:}(e \to \mathbb{N}) \to \mathbb{Q}, \\ \tau(\text{the total number of}) &:= \lambda P^{\mathbb{Q}\to\mathbb{N}}.\mathbf{sum}(P){:}(e \to \mathbb{N}) \to \mathbb{Q}, \end{aligned}$$

[1] `http://www.w3.org/TR/owl-ref`

COP	Copula, common and proper nouns, negation, universal and existential quantifiers
COP+TV	COP plus transitive verbs
COP+TV+DTV	COP+TV plus ditransitive verbs
COP+Rel	COP plus relative pronouns
Lite-English	Copula, left (positive and negative) universal quantification, left relative pronouns, adjectives, common nouns, indeterminate pronoun ”something” and intransitive and transitive verbs
DL-English	Copula, left (positive and negative) universal quantification, relative pronouns, adjectives, common nouns (of which ”thing”), existential quantifiers, intransitive and transitive verbs, negation and conjunction

COP	$\Phi_l(x) \rightarrow A(x)$ $\Phi_r(x) \rightarrow \Phi_l(x)$	$\forall x(\Phi_l(x) \Rightarrow \pm\Phi_r(x))$ $\exists x(\Phi_l(x) \wedge \Phi_r(x))$	No student failed. Some student failed.
COP+ TV	$\Phi_l(x) \rightarrow A(x)$ $\Phi_r(x) \rightarrow \Phi_l(x) \mid \forall y(A(x) \Rightarrow \pm\psi(x,y))$ $\mid \exists y(A(x) \wedge \psi(x,y))$	$\forall x(\Phi_l(x) \Rightarrow \pm\Phi_r(x))$ $\exists x(\Phi_l(x) \wedge \Phi_r(x))$	No student failed. Some student studies every course.
COP+ TV+ DTV	$\Phi_l(x) \rightarrow A(x)$ $\Phi_{tv}(x) \rightarrow \Phi_l(x) \mid \forall y(A(x) \Rightarrow \pm\psi(x,y))$ $\mid \exists y(A(x) \wedge \psi(x,y))$ $\Phi_{dtv}(x,y) \rightarrow \forall z(A(x) \Rightarrow \pm\chi(x,y,z))$ $\mid \exists z(A(x) \wedge \chi(x,y,z))$ $\Phi_r(x) \rightarrow \Phi_{tv}(x) \mid \forall y(A(x) \Rightarrow \pm\Phi_{dtv}(x,y))$ $\mid \exists y(A(x) \wedge \Phi_{dtv}(x,y))$	$\forall x(\Phi_l(x) \Rightarrow \pm\Phi_r(x))$ $\exists x(\Phi_l(x) \wedge \Phi_r(x))$	Every student gives no credit to some student. Some student borrows some book from some library.
COP+ REL	$\Phi_l(x) \rightarrow A(x) \mid \pm\Phi_l(x) \wedge \pm\Phi_l(x)$ $\Phi_r(x) \rightarrow \Phi_l(x)$	$\forall x(\pm\Phi_l(x) \Rightarrow \pm\Phi_r(x))$ $\exists x(\pm\Phi_l(x) \wedge \pm\Phi_r(x))$	Some student who is not diligent is smart.
DL-Lite	$\Phi_l(x) \rightarrow A(x) \mid \exists y\psi(x,y)$ $\Phi_r(x) \rightarrow \Phi_l(x)$	$\forall x(\Phi_l(x) \Rightarrow \pm\Phi_r(x))$	Everybody who studies something is a student.
$\mathcal{ALCI}$	$\Phi_l(x) \rightarrow A(x) \mid \exists y\psi(x,y)$ $\mid \pm\Phi_l(x) \wedge \pm\Phi_l(x)$ $\mid \exists y\psi(x,y) \wedge \pm\Phi_l(y)$ $\Phi_r(x) \rightarrow \Phi_l(x)$	$\forall x(\pm\Phi_l(x) \Rightarrow \pm\Phi_r(x))$	Every student that is not diligent is not a good student.

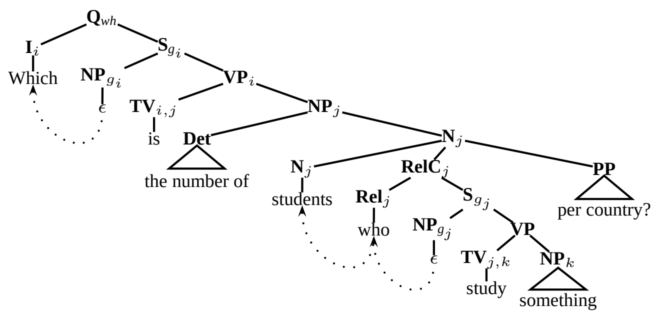

Fig. 1. Top: Coverage of the declarative CLs discussed in this paper and the ontology languages they express. Note that $\psi(x, y)$ (resp. $\chi(x, y, z)$) stands for some binary (resp. ternary) atom. By $\pm$ we convey the fact that an atom or condition may or may not be negated. Complete utterances in these fragments are all of the form **Det N VP**, where **Det** maps, modulo $\tau(\cdot)$, into either $\forall$ or $\exists$, **N** (recursively) into $\Phi_l(x)$, the subject, and **VP** (recursively) into $\Phi_r(x)$, the predicate [14, 17, 4]. Notice that subjects and predicates, but for COP+REL and DL-English, express different properties (they are non-symmetrical). **Below:** `GROUP BY` clauses are captured by **PP** components.

$$\tau(\text{the average number of}) := \lambda P^{\mathbb{Q}\to\mathbb{N}}.\mathbf{avg}(P){:}(e \to \mathbb{N}) \to \mathbb{Q},$$
$$\tau(\text{the number of}) := \lambda P^{e\to\mathbb{N}}.\mathbf{count}(P){:}(e \to \mathbb{N}) \to \mathbb{Q},$$
$$\tau(\text{the number of distinct}) := \lambda P^{e\to\mathbb{N}}.\mathbf{countd}(P){:}(e \to \mathbb{N}) \to \mathbb{Q}.$$

MRs make use of a set $\{e, t, \mathbb{N}, \mathbb{Q}\}$ of basic types. Notice that booleans ($\{0, 1\}$) are positive integers and that as a result bag-typed expressions are implicitly polymorphic. The gammar of ATQ-English is specified as follows. We express `GROUP BY` clauses (followed possibly by a `HAVING` clause) from SQL by means of **PP**s that combine with subordinate clauses. We disregard morphology and polarity issues, which can be easily dealt with by considering *definite clause grammar* (DCG) rules, where parsing is based on SLD-resolution and unification [10].

$\mathbf{Q}_{wh} \to \mathbf{I}_i\ \mathbf{N}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}){:=}\lambda\bar{z}.\tau(\mathbf{I}_i)(\tau(\mathbf{N}_i))(\lambda i.\tau(\mathbf{S}_{g_i}))$		
$\mathbf{Q}_{wh} \to \mathbf{I}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}){:=}\lambda\bar{z}.\tau(\mathbf{I}_i)(\lambda i.\tau(\mathbf{S}_{g_i}?))$		
$\mathbf{Q}_{Y/N} \to$ does $\mathbf{NP}_i\ \mathbf{VP}_i?$	$\tau(\mathbf{Q}_{Y/N}){:=}\tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i))$		
$\mathbf{Q}_{Y/N} \to$ is $\mathbf{NP}_i\ \mathbf{VP}_i?$	$\tau(\mathbf{Q}_{Y/N}){:=}\tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i))$		
$\mathbf{S}_{g_i} \to \mathbf{NP}_{g_i}\ \mathbf{VP}_i$	$\tau(\mathbf{S}_{g_i}){:=}\tau(\mathbf{NP}_{g_i})(\tau(\mathbf{VP}_i))$		
$\mathbf{N}_i \to \mathbf{Adj}\ \mathbf{N}_i$	$\tau(\mathbf{N}_i){:=}\tau(\mathbf{Adj})(\tau(\mathbf{N}_i))$		
$\mathbf{N}_i \to \mathbf{N}_i\ \mathbf{RelC}_i\ \mathbf{PP}$	$\tau(\mathbf{N}_i){:=}\tau(\mathbf{Rel}_i)(\tau(\mathbf{N}_i))(\tau(\mathbf{PP}))$		
$\mathbf{RelC}_i \to \mathbf{Rel}_i\ \mathbf{S}_{g_i}$	$\tau(\mathbf{RelC}_i){:=}\tau(\mathbf{Rel}_i)(\lambda i.\tau(\mathbf{S}_{g_i}))$	$\mathbf{VP}_i \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_i){:=}\tau(\mathbf{Adj})$
$\mathbf{VP}_i \to \mathbf{VP}_i\ \mathbf{Crd}\ \mathbf{VP}_i$	$\tau(\mathbf{VP}){:=}\tau(\mathbf{Crd})(\tau(\mathbf{VP}_i))(\tau(\mathbf{VP}_i))$	$\mathbf{VP}_i \to$ is a $\mathbf{N}_i$	$\tau(\mathbf{VP}_i){:=}\tau(\mathbf{N}_i)$
$\mathbf{VP}_i \to \mathbf{TV}_{i,j}\ \mathbf{NP}_j$	$\tau(\mathbf{VP}_i){:=}\tau(\mathbf{TV}_{i,j})(\tau(\mathbf{NP}_j))$	$\mathbf{VP}_i \to \mathbf{IV}_i$	$\tau(\mathbf{VP}_i){:=}\tau(\mathbf{IV}_i)$
$\mathbf{NP}_i \to \mathbf{Det}\ \mathbf{N}_i$	$\tau(\mathbf{NP}_i){:=}\tau(\mathbf{Det})(\tau(\mathbf{N}_i))$	$\mathbf{NP}_i \to \mathbf{Pro}_i$	$\tau(\mathbf{NP}_i){:=}\tau(\mathbf{Pro}_i)$
$\mathbf{PP} \to \mathbf{PP}\ \mathbf{RelC}_i$	$\tau(\mathbf{PP}){:=}\tau(\mathbf{PP})(\tau(\mathbf{RelC}_i))$	$\mathbf{NP}_i \to \mathbf{Pn}_i$	$\tau(\mathbf{NP}_i){:=}\tau(\mathbf{Pn}_i)$

Notice that $\bar{z} \subseteq FV(\tau(N_i)) \cup FV(\tau(S_{g_i}))$ (this is the expedient that allows us to capture grouping variables). We say that a condition Φ is *equivalent* to a HOL expression $\alpha = \lambda\bar{x}.\beta{:}\tau$, in symbols $\Phi \equiv \alpha$, when $\Phi = \beta$.

Theorem 1. *ATQ-English expresses ATQs.*

Proof. ($\Rightarrow$) We need to show that for every Wh-question Q in ATQ-English there exists an ATQ q of the form s.t. $\tau(Q) \equiv q$. Questions Q come in three kinds, *(i)* aggregate Wh-questions, *(ii)* non-aggregate Wh-questions and *(iii)* (non-aggregate) Y/N-questions. To prove this result, we prove something more general, namely that for each (recursive) $\mathbf{N}$ and/or $\mathbf{VP}$ constituent of ATQ-English, there exists a tree-shaped condition $\Phi(x)$ s.t. they map to, modulo $\tau(\cdot)$, $\lambda x.\Phi(x){:}\ e \to \mathbb{N}$. This we can prove by an easy induction on $\mathbf{VP}$s and $\mathbf{N}$s, taking care that types, polarity and morphosyntactic features, unify. For simplicity, we disregard gap-filler indexes. It is then easy to see that, for instance, ”which is **Det** **N** per **N**” maps to $\lambda x^e.\lambda n^{\mathbb{Q}}.n \approx \alpha(\lambda y^e.\Phi(y) \wedge \Phi'(y)){:}e \to (\mathbb{Q} \to t)$, that ”does **NP** **VP**” maps to $\exists x\Phi(x){:}\ t$, or that ”which **N** **VP**” maps to $\lambda x^e.\Phi(x){:}e \to \mathbb{N}$.

($\Leftarrow$) We need to show that for each ATQ q there exists a question Q in ATQ-English s.t. $\tau(Q) \equiv q$. In order to prove this, we prove, by induction on tree-shaped conditions $\Phi(x)$ rooted in x, that there exists either a **N** or a **VP** constituent in ATQ-English s.t. $\tau(\mathbf{N}) \equiv \Phi(x)$ (resp. $\tau(\mathbf{VP}) \equiv \Phi(x)$):

1. (Basis) If $\Phi(x) = A(x)$, then it has as preimage the **N** A or the **VP** ”is a A”, while if $\Phi(x) = \exists y(R(x, y))$, it has as preimage the **N** or **VP** ”Rs something”.
2. (Inductive step) If $\Phi(x) = \Phi'(x) \wedge \Phi''(x)$, by IH $\Phi'(x)$ is the image of some **N** or **VP** and similarly for $\Phi''(x)$. Hence, $\Phi(x)$ has as preimage either ”**N** **RelC**” (where, e.g., **RelC** rewrites into the**VP** associated to $\Phi''(x)$) or ”**VP** and **VP**’”. The argument is similar for $\Phi(x) = \exists yR(x, y) \wedge \Phi'(y)$.

Clearly then, the ATQ $q(x, \alpha(x)) \leftarrow \Phi(x) \wedge R(x, y) \wedge \Phi'(y)$ (or, more precisely, its equivalent HOL MR) will have as preimage in ATQ English the question ”which is **Det** **N** per **N**’?”, where **Det** is an aggregate determiner. On the other hand, $q(x) \leftarrow \Phi(x)$ will be the image of ”what/who **VP**?” and $q() \leftarrow \exists x \Phi(x)$ will be the image of ”does anybody **VP**?” or ”is anybody **VP**?”. $\square$

Example 2. Consider the following Wh-question:

Which is the number of students who study something, per country? (Q0)

It gives rise to the **count** aggregate query q_0 of Example 1 via the parse tree from Figure 1, which is generated by combining our grammar with the following lexicon. Aggregate determiners express the definite **NP** ”the number of **N**” while the *grouping complement* (a **PP** attachment) ”per **N**” expresses grouping:

– τ(which):=$\lambda R^{\mathbb{Q}\rightarrow(\mathbb{Q}\rightarrow t)}.\lambda n^{\mathbb{Q}}.R(n, m)$:$(\mathbb{Q} \rightarrow (\mathbb{Q} \rightarrow t)) \rightarrow (\mathbb{Q} \rightarrow t)$.
– τ(is):=$\lambda n^{\mathbb{Q}}.\lambda m^{\mathbb{Q}}.n \approx m$:$\mathbb{Q} \rightarrow (\mathbb{Q} \rightarrow t)$.
– $\tau(\epsilon)$:=$\lambda P^{\mathbb{Q}\rightarrow t}.P(n)$: $(\mathbb{Q} \rightarrow t) \rightarrow t$.
– τ(students):=$\lambda x^e.$*Student*(x):$e \rightarrow \mathbb{N}$.
– τ(something):=$\lambda P^{e\rightarrow t}.\exists y^e P(y)$:$(e \rightarrow \mathbb{N}) \rightarrow t$
– τ(who):=$\lambda P^{e\rightarrow\mathbb{N}} \lambda Q^{e\rightarrow\mathbb{N}}.\lambda x^e.(P(x) \wedge Q(x))$:$(e \rightarrow \mathbb{N}) \rightarrow ((e \rightarrow \mathbb{N}) \rightarrow (e \rightarrow \mathbb{N}))$.
– τ(per country):=$\lambda P^{e\rightarrow\mathbb{N}}.\lambda y^e.(P(y) \wedge$ *Country*$(z) \wedge$ *comesFrom*$(y, z))$:$(e \rightarrow \mathbb{N}) \rightarrow (e \rightarrow \mathbb{N})$.
– τ(per country):=$\lambda Q^{e\rightarrow\mathbb{N}}.\lambda P^{e\rightarrow\mathbb{N}} \lambda y^e.(P(y) \wedge$ *Country*$(z) \wedge$ *comesFrom*$(y, z) \wedge Q(z))$: $(e \rightarrow \mathbb{N}) \rightarrow ((e \rightarrow \mathbb{N}) \rightarrow (e \rightarrow \mathbb{N}))$.
– τ(study):=$\lambda \alpha^{(e\rightarrow\mathbb{N})\rightarrow\mathbb{N}}.\lambda x^e.\alpha(\lambda y^e.$*takesCourse*$(x, y))$:$((e \rightarrow \mathbb{N}) \rightarrow \mathbb{N}) \rightarrow (e \rightarrow \mathbb{N})$.

The value of $\tau(\cdot)$ on the whole question (after λ-application and abstraction and β-normalization) is

$$\lambda x^e.\lambda m^{\mathbb{Q}}.m \approx \mathbf{count}(\lambda y^e.\mathit{Student}(y) \wedge \exists z^e(\mathit{takesCourse}(y, z)) \wedge \\ \mathit{comesFrom}(y, x) \wedge \mathit{Country}(x)) \colon e \rightarrow (\mathbb{Q} \rightarrow t),$$

i.e., the value of $\tau(\cdot)$ on the (root) component $\mathbf{Q}_{wh}$ (see again Figure 1). Clearly, $\tau(\text{Q0}) \equiv q_0$. ▮

Expressing Comparisons, $\vee$, $\neg$ and $\forall$. By covering comparative (both majorative and diminutive) and equative adjectives we can capture comparisons, i.e., the constants $\theta \in \{\leq, \geq, <, >, \approx\}$ of type $\mathbb{Q} \rightarrow (\mathbb{Q} \rightarrow t)$ over the rationals $\mathbb{Q}$ (which we assume to be totally ordered) and comparison atoms $t\,\theta\,t'$ [1]. Another way of increasing the coverage of our CL consists in considering all the quantifiers and boolean operators definable on conditions, viz., $\vee$, $\forall$ and $\neg$ (disjunction is equivalent to the `UNION` and `UNION ALL` query constructors in SQL). Thus doing we express:

1. Tree-shaped conditions with comparisons, $t\,\theta\,t'$ ($\leq$-ATQs).
2. Negations (with e.g. ”does not”) of tree-shaped conditions, $\neg\Phi(x)$ ($\neg$-ATQs).
3. Unions (with ”or”) of tree-shaped conditions, $\Phi(x) \vee \Phi'(x)$ ($\vee$-ATQs).
4. Universal (with ”only”) tree-shaped conditions, $\forall x(R(x, y) \Rightarrow \Phi(y))$ ($\forall$-ATQs).

Example 3. Given the **Adj** ”heavy”, put: *(i) heavier*(x,y):=$\exists n \exists n'($*hasCredits*$(x,n) \wedge$ *hasCredits*$(y,n') \wedge n \geq n')$, *(ii) s-heavier*(x,y):=$\exists n \exists n'($*hasCredits*$(x,n) \wedge$*hasCredits*(y,n') $\wedge\, n > n')$ and *(iii) as-heavy*(x,y):=$\exists n \exists n'($*hasCredits*$(x,n) \wedge$ *hasCredits*$(y,n') \wedge n \approx n')$ of type $e \rightarrow (e \rightarrow t)$, where *hasCredits* is an expression of type $e \rightarrow (\mathbb{Q} \rightarrow t)$. Next, we create lexical entries for ”is heavier than”, ”is strictly heavier than” (the majoratives) and for ”is as heavy as” (the equative), as well as entries for the logical operations:

– τ(is heavier than):=$\lambda\alpha^{(e\rightarrow\mathbb{N})\rightarrow t}.\lambda x^e.\alpha(\lambda y^e.$*higher*$(x,y))$:$((e\rightarrow\mathbb{N})\rightarrow t)\rightarrow(e\rightarrow t)$.
– τ(is as heavy as):=$\lambda\alpha^{(e\rightarrow\mathbb{N})\rightarrow t}.\lambda x^e.\alpha(\lambda y^e.$*as-high*$(x,y))$:$((e\rightarrow\mathbb{N})\rightarrow t)\rightarrow(e\rightarrow t)$.
– τ(is strictly heavier than):=$\lambda\alpha^{(e\rightarrow\mathbb{N})\rightarrow t}.\lambda x^e.\alpha(\lambda y^e.$*s-higher*$(x,y))$:$((e\rightarrow\mathbb{N})\rightarrow t)$ $\rightarrow(e\rightarrow t)$.
– τ(not):=$\lambda P.\neg P$:$(e\rightarrow\mathbb{N}) \rightarrow (e\rightarrow\mathbb{N})$.
– τ(or):=$\lambda P^{e\rightarrow\mathbb{N}}.\lambda Q^{e\rightarrow\mathbb{N}}.\lambda x^e.(P(x) \vee Q(x))$: $(e\rightarrow\mathbb{N})\rightarrow((e\rightarrow\mathbb{N})\rightarrow t)$
– τ(only):=$\lambda Q.\lambda P.\forall x(P(x) \Rightarrow Q(x))$:$(e\rightarrow\mathbb{N})\rightarrow((e\rightarrow\mathbb{N})\rightarrow t)$.
– τ(who):=$\lambda P^{e\rightarrow\mathbb{N}}.\lambda x^e.P(x)$:$(e\rightarrow\mathbb{N})\rightarrow(e\rightarrow\mathbb{N})$.
– τ(some):=$\lambda P^{e\rightarrow\mathbb{N}}.\lambda Q^{e\rightarrow\mathbb{N}}.\exists x^e(P(x) \wedge Q(x))$: $(e\rightarrow\mathbb{N})\rightarrow((e\rightarrow\mathbb{N})\rightarrow t)$.

Content words can be easily added as follows. **TV**s like ”comes from” give way to the entry τ(comes from):=$\lambda\alpha^{(e\rightarrow\mathbb{N})\rightarrow t}.\lambda x^e.\alpha(\lambda y^e.$*comesFrom*$(x,y))$:$((e\rightarrow\mathbb{N})\rightarrow t)\rightarrow$ $(e\rightarrow t)$ and **N**s such as ”student”, to entries such as τ(student):=$\lambda x^e.$*Student*(x):$e\rightarrow\mathbb{N}$. Consider now the following controlled Wh-questions:

Which course is heavier than (strictly heavier than, as heavy as) some course? (Q1)

Who is a student who does not come from Italy? (Q2)

Who is a student or comes from some country? (Q3)

Which student studies only courses held in universities? (Q4)

They can now be successfully parsed. ATQ-English can now express simple queries with $\forall, \neg, \vee$ and $\theta \in \{\leq, \geq, \approx, <, >\}$. ∎

4 Data Complexity of QA

In this section we show that adding $\forall$, $\leq$ and $\neg$ to conditions make query answering hard. The $\vee$ operator alone, however, need not [6, 1]. Modulo $\tau(\cdot)$, reasoning over CLs is polynomially equivalent (in data complexity) to reasoning over their MRs [14]. In what follows, we identify CLs with their MRs (i.e., with constraints and/or formal queries) and reason solely on these MRs.

To check whether a tuple (c,n) is a certain answer for ATQ $q(x,\alpha(y)) \leftarrow \Phi$ to an OBDAS $(\mathcal{O},D)$, in general, we *(i)* check whether x is instantiated to c by a certain assignment and then *(ii)* loop over the (finitely many) certain assignments for y, updating at each step the value of α on the group H_c, until α returns n. Otherwise, our procedure will return a negative answer. In other words the data complexity of answering an ATQ depends, ultimately, on that of computing $Sat_D^{\mathcal{O}}(\Phi)$ and coincides, for this reason, with the data complexity of answering its core (i.e., with that of answering CQs over OBDASs), whenever this data complexity is known.

Theorem 2. *Answering ATQs (and unions thereof) w.r.t. Lite-English and COP declarations is in* **LogSpace** *in data complexity.*

Proof. Lite-English expresses the *DL-Lite* ontology language [4]. It moreover, contains COP. The result follows immediately from the data complexity of answering CQs over *DL-Lite* OBDASs [6]. $\square$

Theorem 3. *Answering ATQs is* **coNP***-complete for COP+Rel and DL-English.*

Proof. (Sketch) It can be shown that QA for ATQs and COP+Rel is **coNP**-hard in data complexity. On the other hand, DL-English contains COP+Rel and is contained in the two-variable fragment of FO. The same holds for the cores of ATQs. Hence, data complexity is in **coNP** [13]. $\square$

Theorem 4. *Answering* $\forall$*-ATQs (and unions thereof) over OBDASs* $(\mathcal{O}, D)$ *where* $\mathcal{O}$ *is a COP ontology is* **coNP***-hard in data complexity. It is in* **coNP** *for DL-English, COP+TV, Lite-English and COP.*

Proof. (Hardness.) By reduction from the **NP**-complete satisfiability problem for *2+2 clauses* (2+2-SAT), where, given a conjunction $\phi := \psi_1 \wedge ... \wedge \psi_k$ of k propositional clauses of the form $\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$, we ask whether there exists a truth assignment (ta) $\delta(\cdot)$ s.t. $\delta(\phi) = 1$. 2+2-SAT was shown to be **NP**-complete by [15], whose prove we adapt.

Encode ϕ into a DB D_ϕ as follows. Consider the predicates P_1, P_2, N_1 and N_2, and for each ψ_i, set: $\{(i, p_{i1})\} \subseteq P_1^{D_\phi}$, $\{(i, p_{i2})\} \subseteq P_2^{D_\phi}$, $\{(i, n_{i1})\} \subseteq N_1^{D_\phi}$ and $\{(i, n_{i2})\} \subseteq N_2^{D_\phi}$. Next, consider three predicates A_f and A_t (unary), and *Val* (binary). Set $\mathcal{O} := \{\forall x(A_f(x) \Rightarrow \neg A_t(x))\}$ and put $\top \in A_t^{D_\phi}$. Finally, consider the following (boolean) $\forall$-ATQ $q() \leftarrow \Phi$ whose body is defined as follows, $\Phi := \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (P_1(x, y_1) \wedge \forall z_1(\mathit{Val}(y_1, z_1) \Rightarrow A_f(z_1)) \wedge P_2(x, y_2) \wedge \forall z_2(\mathit{Val}(y_2, z_2) \Rightarrow A_f(z_2)) \wedge N_1(x, y_3) \wedge \exists z_2(\mathit{Val}(y_3, z_3) \wedge A_t(z_3)) \wedge N_2(x, y_3) \wedge \exists z_3(\mathit{Val}(y_4, z_4) \wedge A_t(z_4)))$. We claim that

$$() \notin \mathit{cert}(q, \mathcal{O}, D_\phi) \text{ iff } \phi \text{ is satisfiable.}$$

($\Leftarrow$) If ϕ is satisfiable, then there exists a ta $\delta(\cdot)$ s.t. $\delta(\phi) = 1$. Define a DB D extending D_ϕ and that is a model of $\mathcal{O}$ as follows. Pick a $v \in \Delta$ and put, for all p, $(p, v) \in \mathit{hasValue}^D$ and $v \in A_t^D$ iff $\delta(p) = 1$. Clearly, D is as desired and, for all γ, $D, \gamma \not\models \Phi$, i.e., $\mathit{Sat}_D(\Phi)$ is empty. Thus, $\mathit{Sat}^{\mathcal{O}}_{D_\phi}(\Phi)$ is empty and the result follows.

($\Rightarrow$) If the certain answers are empty, then there exists a DB D s.t., for all γ, $D, \gamma \not\models \Phi$. Define now a tvd $\delta(\cdot)$ from the p_{ij}s and the n_{ij}s, to $\{0, 1\}$, by putting, for all such propositional atoms p, $\delta(p) = 1$ iff there exists a v s.t. $(p, v) \in \mathit{hasValue}^D$ and $v \in A_t^D$. Clearly, $\delta(\phi) = 1$.

(Membership.) For the upper bound, we remind the reader that the core $\breve{q}$ (and body Φ) of an ATQ q are formulas from the two variable fragment of FO, for which data complexity is in $\mathbf{coNP}$ [13]. $\square$

Theorem 5. *Answering* $\neg$*-ATQs (and unions thereof) over OBDASs* $(\mathcal{O}, D)$ *where* $\mathcal{O}$ *is an empty ontology is* **coNP***-hard in data complexity. It is in* **coNP** *for DL-English, COP+TV, Lite-English and COP.*

Proof. (Sketch) By reduction, again from 2+2-SAT. The proof is also a variation of the previous two. We put $\mathcal{O} := \emptyset$, leave D_ϕ unchanged and consider the boolean query with body $\Phi := \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (P_1(x, y_1) \wedge \neg A_t(y_1) \wedge P_2(x, y_2) \wedge \neg A_t(y_2) \wedge N_1(x, y_3) \wedge A_t(y_3) \wedge N_2(x, y_3) \wedge A_t(y_4))$. The intuition is that a 2+2 clause propositional atom p is true under tvd $\delta(\cdot)$ iff $p \in A_t^D$ holds in DB D.

For the upper bound we reason as previously, by observing that the cores $\breve{q}$ (and bodies Φ) of a $\neg$-ATQs q are also contained in the two-variable fragment of FO. □

Theorem 6. *Answering $\leq$-ATQs (and unions thereof) over OBDASs $(\mathcal{O}, D)$ where $\mathcal{O}$ is an empty ontology is* **coNP***-hard in data complexity.*

Proof. (Sketch) The lower bound is obtained by reduction, again, from 2+2-SAT. The proof is a slight variation of the previous one. Notice that $\leq$ can be used to simulate negation. $\mathcal{O}$ and D_ϕ stay unchanged, and we consider the boolean CQ of (boolean tree-shaped body) $\Phi := \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (P_1(x, y_1) \wedge y_1 \leq 0 \wedge P_2(x, y_2) \wedge y_2 \leq 0 \wedge N_1(x, y_3) \wedge y_3 > 0 \wedge N_2(x, y_3) \wedge y_4 > 0)$. The intuition is that a 2+2 clause literal p is true under this encoding iff the (ground FO) fact $p > 0$ is true. □

	$\vee$-**ATQ**s	$\leq$-**ATQ**s	$\forall$-**ATQ**s	$\neg$-**ATQ**s
Lite-English	in **LogSpace**	**coNP**-hard	**coNP**-complete	**coNP**-complete
COP	in **LogSpace**	**coNP**-hard	**coNP**-complete	**coNP**-complete
COP+TV	(unknown)	**coNP**-hard	**coNP**-hard	**coNP**-hard
COP+TV+DTV	(unknown)	**coNP**-hard	**coNP**-hard	**coNP**-hard
DL-English	**coNP**-complete	**coNP**-hard	**coNP**-complete	**coNP**-complete
COP+Rel	**coNP**-complete	**coNP**-hard	**coNP**-complete	**coNP**-complete

5 Conclusions

We have proposed a class of aggregate queries, viz., tree-shaped aggregate queries (ATQs), equipped with a certain answers semantics. ATQs can be considered a subclass of the so-called epistemic aggregate queries defined in [7].

We have shown how to express ATQs in controlled English by means of the CL ATQ-English. We analyse `GROUP BY` clauses as modifiers of the question's subject (i.e., its subject **N** constituent). By using higher order logic (HOL) and, hence, (bag) typed, intermediate semantic representations, we ensure that the translation $\tau(\cdot)$ is compositional and that query answering (QA) with ATQ-English questions reduces (w.r.t. data complexity) to QA with ATQs.

We have shown that answering ATQ-English controlled aggregate questions over declarative CLs such as ACE-OWL and other declarative languages that overlap in expressiveness with ACE-OWL reduces to conjunctive query answering. Therefore, computing aggregates does not have any significant impact on data complexity. Aggregates by themselves are constructs that any CL interface to OBDASs can support. What does have an impact are query conditions, alone or in combination with expressive ontology languages.

We have also shown that *(i)* allowing for full boolean operations in the declarative CL/ontology language and/or *(ii)* full boolean operations ($\forall, \neg$) and/or $\leq$ in the interrogative CL/query language, yields immediately intractability.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Welsey, 1995.
2. I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.
3. Franz Baader, Diego Calvanese, Daniele Nardi, Peter Patel-Schneider, and Deborah McGuinness. *The Description Logic Handbook*. Cambridge University Press, 2003.
4. Raffaella Bernardi, Diego Calvanese, and Camilo Thorne. Lite natural language. In *IWCS-7. Proceedings of the 7th International Workshop on Computational Semantics*, 2007.
5. Abraham Bernstein, Esther Kaufman, Anne Göring, and Christoph Kiefer. Querying ontologies: A controlled english interface for end-users. In *ISWC 2005. Proceedings of the 4th International Semantic Web Conference*, 2005.
6. Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *KR 2006. Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning*, 2006.
7. Diego Calvanese, Werner Nutt, Evgeny Kharlamov, and Camilo Thorne. Aggregate queries over ontologies. In *ONISW 2008. Proceedings 2nd International Workshop on Ontologies and Information Systems for the Semantic Web*, 2008.
8. James Clifford. Natural language querying of historical databases. *Computational Linguistics*, 14(4):10–35, 1988.
9. Norbert E. Fuchs and Kaarel Kaljurand. Mapping Attempto Controlled English to OWL DL. In *ESWC 2006. Demos and Posters of the 3rd European Semantic Web Conference*, 2006.
10. Daniel Jurafsky and James Martin. *Speech and Language Processing*. Prentice Hall, 2000.
11. Sela Mador-Haim, Yoad Winter, and Anthony Braun. Controlled language for geographical information system queries. In *ICos5. Proceedings of the 5th International Workshop on Inference in Computational Semantics*, 2006.
12. Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.
13. Ian Pratt. Data complexity of the two-variable fragment with counting quantifiers. *Information and Computation*, 207(8):867–888, 2009.
14. Ian Pratt and Allan Third. More fragments of language. *Notre Dame Journal of Formal Logic*, 47(2):151–177, 2006.
15. Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278, 1993.
16. Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.
17. Camilo Thorne and Diego Calvanese. Exploring ontology-based data access. In *CNL 2009. Proceedings of the Workshop on Controlled Natural Language*, 2009.
18. Moshe Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, 1982.