EXPRESSING CONJUNCTIVE AND AGGREGATE QUERIES OVER ONTOLOGIES WITH CONTROLLED ENGLISH

Camilo Thorne

Free University of Bozen-Bolzano

cthorne@inf.unibz.it

Abstract. We propose to characterize the computational complexity of answering questions in ontology-mediated controlled language interfaces to structured data sources by *expressing* ontology-based data access in controlled English. This means: compositionally mapping a controlled subset of English into knowledge bases and formal queries for which the computational complexity of ontology-based data access is known. In the present paper, we extend this approach to conjunctive queries and to conjunctive queries with aggregation functions.

1. Introduction

Lately, there has been a renewed interest within the computational linguistics community (Minock 2005, Lesmo & Robaldo 2007) in natural language interfaces to databases (NLIDBs), where what is aimed at is managing, with natural language (NL), relational databases (DBs). In particular, robust interfaces supporting *controlled fragments* (CLs) of English and based on ontologies, computational semantics and deep semantic parsing have been developed, by, for instance, the Attempto project (Bernstein, et al. 2003, Fuchs, et al. 2005). Controlled languages are fragments of NL tailored to fit data management tasks by, typically, constraining their restricted vocabulary (and syntax), thereby stripping them from ambiguity, whether structural or semantic. Controlled languages allow a trade-off between the coverage and the accuracy of the translation of questions into formal queries. Ontologies (the conceptualizations of the domain) play the intermediate role between the CL's vocabulary and the domain terminology.

However, some important issues regarding controlled English interfaces have not been, to the best of our knowledge, fully adressed. One of them is the tractability and untractability of processing CL information requests and utterances, viz., how difficult is declaring and accessing structured data with a controlled English interface? And by difficult, we mean its *computational complexity*. We believe that a way of adressing this issue consists in *expressing ontology based data access with CLs*. By this we mean designing declarative and interrogative controlled subsets of English that compositionally map through a semantic mapping $[\![.]\!]$ (taken from NL formal semantics) into formal queries, ontologies and database facts, their meaning representations (MRs). Ontology based data access provides the logical underpinning of accessing structured data w.r.t. ontologies and its computational complexity, a measure of how difficult a task it might be.

The main purpose of this paper is twofold. On the one hand, we will say what means to express in CL ontology based data access. On the other hand, we will proceed to express in controlled English a class of formal queries known as *conjunctive queries*. Conjunctive queries are good in that with them we reach an optimal computational complexity. Last, but not least, we will extend our controlled language to cover aggregate queries, which are conjunctive queries to which the basic SQL aggregation functions, COUNT, MIN, MAX and SUM, have been added.

2. Ontology Based Data Access

Accessing and declaring data w.r.t. an ontology or conceptualization can be characterized in terms of formal logic as follows (Rosati 2007). A *relational query* q of arity n is a formal expression $q(\overline{\mathbf{x}}) \leftarrow \mathbf{Q}\overline{\mathbf{y}}\beta(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, where $q(\overline{\mathbf{x}})$ is the *head* and $\overline{\mathbf{x}}$ denotes a sequence of n variables, the query's *distinguished variables*, and $\mathbf{Q}\overline{\mathbf{y}}\beta(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ is the *body*, a first order logic (FOL) quantified boolean combination of relational atoms where the distinguished variables occur free and the others (the sequence $\overline{\mathbf{y}}$) bound to a quantifier. $\mathbf{Q}\overline{\mathbf{y}}$ denotes the sequence of its quantifier prefixes. When no confusion arises, we shall abbreviate $\mathbf{Q}\overline{\mathbf{y}}\beta(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ with $\Phi[\overline{\mathbf{x}}]$. A query is said to be *boolean* if its arity is $n = 0$. A collection of such queries is called a *query language*. A *relational database* (DB) $\mathcal{D}$ is a finite set of ground atoms over a *schema* $\mathbf{R} := \{R_1, ..., R_n\}$, where, for $i \in [1, n]$, R_i is a relation symbol of arity $m \geq 1$, and over a countably infinite *domain* **Dom** of constants. The *active domain* $adom(\mathcal{D})$ of $\mathcal{D}$ is the set of constants that occur in $\mathcal{D}$ (a finite subset of **Dom**). An *ontology* $\mathcal{O}$ is a set of FOL axioms that make explicit a certain number of constraints holding over a domain. They are typically defined over some fragment of FOL called an *ontology language*. This language should be rich enough to express DBs (i.e., DB atoms). The pair $\langle \mathcal{O}, \mathcal{D} \rangle$ is called a *knowledge base* (KB), and can be seen as a FOL logical theory: a set of ground atoms (the DB) plus a set of axioms (the ontology). A *gound substitution* is a function $\sigma(.)$ from $Var(q)$, the set of variables of q, into **Dom**. They are extended to sequences of variables in the standard way. KBs and substitutons give rise to the *certain answers semantics* of query q of arity n over a KB $\langle \mathcal{O}, \mathcal{D} \rangle$, denoted $q(\langle \mathcal{O}, \mathcal{D} \rangle)$. It consists in collecting the values in $adom(\mathcal{D})$ of all the ground substitutions $\sigma(.)$ for which $\langle \mathcal{O}, \mathcal{D} \rangle$ *logically entails* $q\sigma$, where $q\sigma$ denotes the grounding of q by $\sigma(.)$. Formally, $q(\langle \mathcal{O}, \mathcal{D} \rangle) := \{\sigma(\overline{\mathbf{x}}) \in adom(\mathcal{D})^n \mid \sigma \text{ s.t. } \langle \mathcal{O}, \mathcal{D} \rangle \models q\sigma\}$. To investigate its computational complexity we must look at the associated *recognition problem*:

Definition 1. (QA) The KB *query answering* (QA) decision problem is the FOL entailment problem stated as follows: given a KB $\langle \mathcal{O}, \mathcal{D} \rangle$, a sequence $\overline{\mathbf{c}} \in \mathbf{Dom}^n$ of n constants, a CQ q of arity n and distinguished variables $\overline{\mathbf{x}}$, check if there exists a ground substitution $\sigma(.)$ s.t. $\sigma(\overline{\mathbf{x}}) = \overline{\mathbf{c}}$ and $\langle \mathcal{O}, \mathcal{D} \rangle \models q\sigma$ holds, where $q\sigma$ is the grounding of q by $\sigma(.)$.

When we focus on $\#(adom(\mathcal{D}))$ (the number of constants of $\mathcal{D}$) while considering constant both $size(q)$ (the number of symbols of the query) and $\#(\mathcal{O})$ (the number of axioms), we speak, in a manner set by (Vardi 1982), of the *data complexity* of QA. Such complexity will depend on the query language and the ontology language chosen (Rosati 2007).

The certain answers semantics can provide a formal semantics for ontology mediated CL data access interfaces and QA's data complexity both a measure of their difficulty and a criterion for optimality. To implement this strategy we need, we believe, to go through two stages: **(i)** We need to choose an ontology language and a query language for which the computational complexity of QA is known and for which data complexity is optimal. **(ii)** We need to *express with controlled English* QA.

3. Expressing QA with Controlled English

A *compositional translation* $[\![.]\!]$, as proposed and conceived by Montague in (Montague 1970) is a function that homomorphically maps a fragment of natural language (English

in our case) into, basically, FOL augmented with the types, the lambda abstraction and the function application constructs of the simply typed λ-calculus, a.k.a. λ-FOL. They assign to NL utterances a λ-FOL formula: its *meaning representation* (MR). The key feature of compositional translations is that they can be made to map declarative fragments of NL into ontology languages and interrogative fragments into query languages.

Definition 2. (Expressing QA) Given an ontology language $\mathcal{L}$ and a query language $\mathcal{Q}$, *expressing* QA *in controlled English* consists in: **(i)** Defining a grammar G and a compositional translation $[\![.]\!]$ for a controlled declarative fragment $L(G)$ s.t. $[\![.]\!]$ maps $L(G)$ into $\mathcal{L}$. **(ii)** Defining a grammar G' and a compositional translation $[\![.]\!]$ for a controlled interrogative fragment $L(G')$ s.t. $[\![.]\!]$ maps $L(G')$ into $\mathcal{Q}$.

We have dealt elsewhere with the problem of expressing KBs and ontology languages by expressing, in particular, the DL-Lite$_{R,\sqcap}$ ontology language or logic and, in general, the DL-Lite family of DLs (Calvanese, et al. 2007). Description logics (DLs) are knowledge representation logics that conceptually model a domain in terms of classes, roles (binary relations among classes) and inheritance relations between classes and roles. In (Bernardi, et al. 2007, Thorne 2007) we define a declarative CL, Lite-English, a compositional translation $[\![.]\!]$ and show that:

Theorem 1. (Bernardi et al. 2007) *For every sentence S in the CL Lite-English, there exists a DL-Lite$_{R,\sqcap}$ assertion α s.t. $[\![S]\!] = \alpha$. Conversely, every DL-Lite$_{R,\sqcap}$ assertion α is the image by $[\![.]\!]$ of some sentence S in Lite-English.*

To get the whole picture we need to look now at query languages. It turns out to be that QA for DL-Lite$_{R,\sqcap}$ is optimal w.r.t. data complexity, falling under **LOGSPACE** (actually, $\mathbf{AC^0}$), a minimal complexity class, when we choose as query language the class of relational queries known as *ruled-based conjunctive queries* (CQs). Conjunctive queries are queries over a schema **R** whose body is a conjunction of existentially quantified relational atoms. Expressing query languages w.r.t. which QA's computational complexity is optimal can shed light on the conditions under which the task of accessing data w.r.t. an ontology with CL might be a relatively easy task.

4. Expressing Conjunctive Queries

In this section we will show how to express *graph-shaped simple conjunctive queries*, a subclass of the class of CQs, for which QA is optimal too. A typical boolean graph-shaped query over, say, the constant $Mary$ and the binary predicates $loves$ and $hates$ is

(1) $$q() \leftarrow \exists x \exists y(loves(Mary, x) \wedge hates(x, y))$$

which we would like to express through the CL Y/N-question

(2) Does Mary love somebody who hates somebody?

And a typical non-boolean graph-shaped query over the same set of relational symbols (i.e., the schema $\{loves, hates\}$) is

(3) $$q(x) \leftarrow \exists y(loves(x, y) \wedge hates(x, y))$$

(Lexical rule)	(Value of $[\![.]\!]$ on word and category)
$\mathbf{Det} \rightarrow$ some	$\lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))\colon (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$
$\mathbf{Pro}_i \rightarrow$ somebody	$\lambda P.\exists x P(x)\colon (e \rightarrow t) \rightarrow t$
$\mathbf{Pro}_i^- \rightarrow$ anybody	$\lambda P.\exists x P(x)\colon (e \rightarrow t) \rightarrow t$
$\mathbf{Coord} \rightarrow$ and	$\lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))\colon (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow (e \rightarrow t))$
$\mathbf{Relpro}_i \rightarrow$ who	$\lambda P.\lambda x.P(x)\colon (e \rightarrow t) \rightarrow (e \rightarrow t)$
$\mathbf{Pro}_i \rightarrow$ him	$\lambda P.P(x)\colon (e \rightarrow t) \rightarrow t$
$\mathbf{Pro}_i \rightarrow$ himself	$\lambda P.P(x)\colon (e \rightarrow t) \rightarrow t$
$\mathbf{Intpro} \rightarrow$ which	$\lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)\colon (e \rightarrow t) \rightarrow (e \rightarrow t)$
$\mathbf{Intpro}_i \rightarrow$ who$_i$	$\lambda P.\lambda x.P(x)\colon (e \rightarrow t) \rightarrow (e \rightarrow t)$
$\mathbf{NP}_{gap_i} \rightarrow \epsilon$	$\lambda P.P(x)\colon (e \rightarrow t) \rightarrow t$
$\mathbf{N}_i \rightarrow$ man,...	$\lambda x.man(x)\colon e \rightarrow t$,...
$\mathbf{IV}_i \rightarrow$ runs,...	$\lambda x.run(x)\colon e \rightarrow t$,...
$\mathbf{IV}_i^- \rightarrow$ run,...	$\lambda x.run(x)\colon e \rightarrow t$,...
$\mathbf{TV}_{i,j} \rightarrow$ loves,...	$\lambda\alpha.\lambda x.\alpha(\lambda y.loves(x,y))\colon ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$,...
$\mathbf{TV}_{i,j}^- \rightarrow$ love,...	$\lambda\alpha.\lambda x.\alpha(\lambda y.loves(x,y))\colon ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$,...
$\mathbf{TV}_{i,j}^p \rightarrow$ loved,...	$\lambda\alpha.\lambda x.\alpha(\lambda y.loves(x,y))\colon ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$,...
$\mathbf{Adj}_i \rightarrow$ mortal,...	$\lambda x.mortal(x)\colon e \rightarrow t$,...
$\mathbf{Pn}_i \rightarrow$ Mary,...	$\lambda P.P(Mary)\colon (e \rightarrow t) \rightarrow t$,...

Table 1: Lexical rules for `GCQ-English`.

which we would like to express through the `CL` `Wh`-question (containing an anaphoric pronoun)

(4) Who loves somebody who hates him?

Definition 3. (GCQs) A non-boolean *graph-shaped simple conjunctive query* (`GCQ`) of arity ≤ 1 is a `CQ` over a schema $\mathbf{R}$ composed of relation symbols of arity ≤ 2 of the form $q := q(x) \leftarrow \Phi[x]$ where the body $\Phi[x]$ is inductively defined as:

$$\begin{array}{lcl} \Phi[x] & := & A_{i_0}(x) \wedge ... \wedge A_{i_m}(x) \wedge R_{j_0}(x,x) \wedge ... \wedge R_{j_m}(x,x) \wedge R_{j_0}(x,c) \wedge R_{j_m}(x,c). \\ \Phi[x] & := & \Phi'[x] \wedge \exists y(A_{i_0}(x) \wedge ... \wedge A_{i_m}(x) \wedge R_{j_0}(x,y) \wedge ... \wedge R_{j_m}(x,y) \wedge R_{j_0}(y,x) \wedge \\ & & \wedge R_{j_m}(y,x) \wedge \Phi''[y]). \end{array}$$

Note that we allow in this definition for empty sequences of conjuncts, e.g., $|A_{i_0}(x) \wedge ... \wedge A_{i_m}(x)| \geq 0$ (where $|.|$ is the function that returns the number of predicates in the body of a relational query). A *boolean* `GCQ` is a query of the form $q := q() \leftarrow \exists y \Phi[y]$, where $\Phi[y]$ is the body of a non-boolean `GCQ`.

4.1. Expressing Conjunctive Queries with `GCQ-English`

`GCQ`s are captured by the interrogative `CL` `GCQ-English`. Questions in `GCQ-English` fall under two main classes : **(i)** `Wh`-questions, that will map into non-boolean `GCQ`s and **(ii)** `Y/N`-questions, that will map into boolean `GCQ`s. For simplicity, we assume grammars to be phrase structure grammars augmented with semantic actions. Phrase structure grammars are composed of two sets of rewriting rules: *lexical rules* (a.k.a. lexicons) and *phrase-structure rules*. Table 2 shows the phrase-structure rules of `GCQ-English`'s grammar and Table 1 its lexicon. Moreover, the latter is divided into two sets: a closed set of *function word* rules, that express (at the semantical level) logical operations and connectives, and an open set of *content word* rules (nouns, adjectives, verbs), a feature we convey through dots.

(Rule)	(Semantic Action)
$\mathbf{Q}_{wh} \rightarrow \mathbf{Intpro}\ \mathbf{N}_i\ \mathbf{S}_{gap_i}?$	$[\![\mathbf{Q}_{wh}]\!] := [\![\mathbf{Intpro}]\!]([\![\mathbf{N}_i]\!])([\![\mathbf{S}_{gap_i}]\!])$
$\mathbf{Q}_{wh} \rightarrow \mathbf{Intpro}_i\ \mathbf{S}_{gap_i}?$	$[\![\mathbf{Q}_{wh}]\!] := [\![\mathbf{Intpro}_i]\!]([\![\mathbf{S}_{gap_i}?]\!])$
$\mathbf{Q}_{Y/N} \rightarrow$ does $\mathbf{NP}_i^-\ \mathbf{VP}_i^-?$	$[\![\mathbf{Q}_{Y/N}]\!] := [\![\mathbf{NP}_i^-]\!]([\![\mathbf{VP}_i^-]\!])$
$\mathbf{Q}_{Y/N} \rightarrow$ is $\mathbf{NP}_i\ \mathbf{VP}_i?$	$[\![\mathbf{Q}_{Y/N}]\!] := [\![\mathbf{NP}_i]\!]([\![\mathbf{VP}_i]\!])$
$\mathbf{S}_{gap_i} \rightarrow \mathbf{NP}_{gap_i}\ \mathbf{VP}_i$	$[\![\mathbf{S}_{gap_i}]\!] := [\![\mathbf{NP}_{gap_i}]\!]([\![\mathbf{VP}_i]\!])$
$\mathbf{VP}_i \rightarrow \mathbf{VP}_i\ \mathbf{Coord}\ \mathbf{VP}_i$	$[\![\mathbf{VP}]\!] := [\![\mathbf{Coord}]\!]([\![\mathbf{VP}]\!])([\![\mathbf{VP}]\!])$
$\mathbf{VP}_i^- \rightarrow \mathbf{VP}_i^-\ \mathbf{Coord}\ \mathbf{VP}_i^-$	$[\![\mathbf{VP}_i^-]\!] := [\![\mathbf{Coord}]\!]([\![\mathbf{VP}^i]\!])([\![\mathbf{VP}_i^-]\!])$
$\mathbf{VP}_i \rightarrow \mathbf{TV}_{i,j}\ \mathbf{NP}_j$	$[\![\mathbf{VP}_i]\!] := [\![\mathbf{TV}_{i,j}]\!]([\![\mathbf{NP}_j]\!])$
$\mathbf{VP}_i \rightarrow$ is $\mathbf{Adj}_i$	$[\![\mathbf{VP}_i]\!] := [\![\mathbf{Adj}_i]\!]$
$\mathbf{VP}_i \rightarrow$ is a $\mathbf{N}_i$	$[\![\mathbf{VP}_i]\!] := [\![\mathbf{N}_i]\!]$
$\mathbf{VP}_i^- \rightarrow \mathbf{IV}_i^-$	$[\![\mathbf{VP}_i^-]\!] := [\![\mathbf{IV}_i^-]\!]$
$\mathbf{VP}_i \rightarrow \mathbf{IV}_i$	$[\![\mathbf{VP}_i]\!] := [\![\mathbf{IV}_i]\!]$
$\mathbf{VP}_i^- \rightarrow \mathbf{TV}_{i,j}^-\ \mathbf{NP}_j$	$[\![\mathbf{VP}_i^-]\!] := [\![\mathbf{TV}_{i,j}^-]\!]([\![\mathbf{NP}_j]\!])$
$\mathbf{VP}_i \rightarrow \mathbf{VP}_i^p$	$[\![\mathbf{VP}_i]\!] := [\![\mathbf{VP}_i^p]\!]$
$\mathbf{VP}_i^p \rightarrow \mathbf{TV}_{i,j}^p\ \mathbf{NP}_j$	$[\![\mathbf{VP}_i^p]\!] := [\![\mathbf{TV}_{i,j}^p]\!]([\![\mathbf{NP}_j]\!])$
$\mathbf{NP}_i^- \rightarrow \mathbf{Det}^-\ \mathbf{N}_i$	$[\![\mathbf{NP}_i^-]\!] := [\![\mathbf{Det}^-]\!]([\![\mathbf{N}_i]\!])$
$\mathbf{NP}_i \rightarrow \mathbf{Pro}_i$	$[\![\mathbf{NP}_i]\!] := [\![\mathbf{Pro}_i]\!]$
$\mathbf{NP}_i \rightarrow \mathbf{Det}\ \mathbf{N}_i$	$[\![\mathbf{NP}_i]\!] := [\![\mathbf{Det}]\!]([\![\mathbf{N}]\!])$
$\mathbf{NP}_i \rightarrow \mathbf{Pn}_i$	$[\![\mathbf{NP}_i]\!] := [\![\mathbf{Pn}_i]\!]$
$\mathbf{NP}_i \rightarrow \mathbf{Pro}_i$	$[\![\mathbf{NP}_i]\!] := [\![\mathbf{Pro}_i]\!]$
$\mathbf{N}_i \rightarrow \mathbf{Adj}\ \mathbf{N}_i$	$[\![\mathbf{N}_i]\!] := [\![\mathbf{Adj}]\!]([\![\mathbf{N}_i]\!])$
$\mathbf{N}_i \rightarrow \mathbf{N}_i\ \mathbf{Relpro}_i\ \mathbf{S}_{gap_i}$	$[\![\mathbf{N}_i]\!] := [\![\mathbf{Relpro}_i]\!]([\![\mathbf{N}_i]\!])([\![\mathbf{S}_{gap_i}]\!]))$

Table 2: Phrase structure rules for `GCQ-English`.

The empty expression ϵ is what in linguistic theory is called a *trace*, a placeholder for the antecedent of the relative pronoun. Symbols occurring in the phrase-structure rewriting rules are called *components* and represent the syntactic chunks into which sentences can be analysed. Symbols that rewrite into words, that is, symbols in the lexicon, are called *categories* or *terminal components* and represent parts of speech, that is, verbs, common and proper nouns, pronouns, adjectives, etc. Some basic morpho-syntactic and semantic features are attached to (some) components. The feature $.^-$ means that the component is of negative polarity, the feature $.^p$, associated to verbs and verb phrase components, indicates that such component is to be inflected in the passive voice. Absence of features indicates that components are in positive polarity and verbs and verb phrases in the active voice. Furthermore, indexes are assigned to components following the standard set by (Pratt 2001) to: **(i)** Resolve intrasentential anaphora: anaphoric pronouns ("him", "himself") resolve with their nearest (antecedent) head noun. **(ii)** Indicate gap-filler dependencies. For simplicity, verbs are in 3rd person singular and in present tense.

A quick glace at the grammar rules of `GCQ-English` will convince the reader that, for instance, the (English) question

(5) Does John love Mary?

and the question

(6) Which man is mortal and loves somebody who hates him?

lie within `GCQ-English`. By the same token, it is easy to see that the question

(7) *Which teacher gives a lesson to his pupils?

lies outside this CL. Why? Because we have no possesive adjectives (e.g., "his") and no ditransitive verbs (e.g., "gives").

Semantic actions mean that we define the translation $[\![.]\!]$ by recursion over the syntactic components of GCQ−English in such a way that the application of each grammar rule, lexical or otherwise, "triggers" $[\![.]\!]$ (Jurafsky & Martin 2000). The intermediate values of this function are called *partial* MRs. When we reach in a Wh-question the $\mathbf{Q}_{wh}$ component $[\![.]\!]$ will map the λ-FOL expression obtained, of the form $[\![\mathbf{Q}_{wh}]\!] = \lambda x.\Phi[x]\colon e \rightarrow t$, into the GCQ $q(x) \leftarrow \Phi[x]$, where $\Phi[x]$ denotes a conjunction of existentially quantified atoms where variable x occurs free. In the case of a Y/N-question, the λ-FOL $[\![\mathbf{Q}_{Y/N}]\!] = \Phi\colon t$ will be mapped into the boolean GCQ $q() \leftarrow \Phi$, where Φ stands for a conjunction of existentially quantified atoms with no free variables. Types ensure that $[\![.]\!]$ always terminates. We can compute, given a GCQ−English question Q, $[\![Q]\!]$ as follows: **(i)** We compute the parse tree of Q. **(ii)** We compute $[\![Q]\!]$ bottom-up, from leaves to root, as in Figure 1. We start by assigning a λ-expression to the leaves. Then, at each internal node, we unify types and compute the λ-application and the β-reduction of its siblings. We omit types for reasons of space. In the end we obtain, at the root of the tree, a GCQ. The circle delimits an island; the dotted line, a gap-filler dependency forced upon by the use of the pronoun.

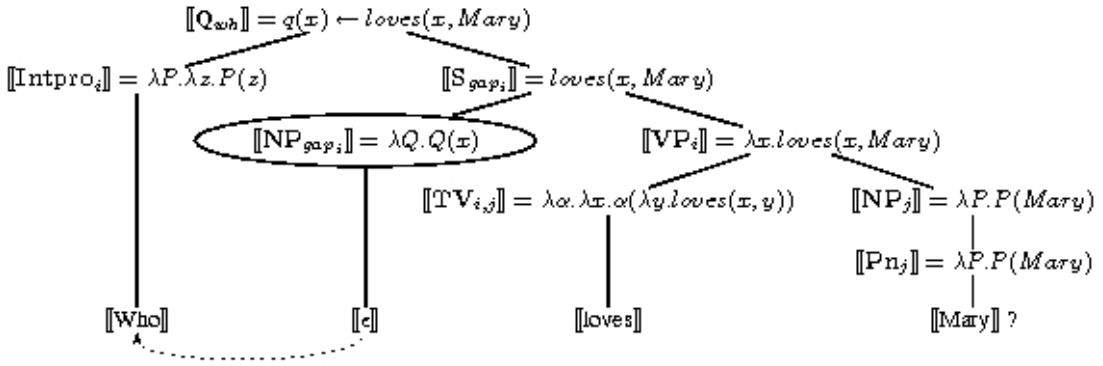

Figure 1: Translating "Who loves Mary?".

Lemma 1. (Expressing GCQs) *For every question Q in GCQ−English, there exists a GCQ q s.t. $[\![Q]\!] = q$. Conversely, every GCQ q is the image by $[\![.]\!]$ of some question Q in GCQ−English.*

Proof. (Sketch) We prove each implication separately:

($\Rightarrow$) We need to show that for every Wh-question Q in GCQ−English there exists a GCQ q of distinguished variable x and body $\Phi[x]$ s.t. $[\![Q]\!] = q(x) \leftarrow \Phi[x]$. Given that the only recursive components in GCQ−English's grammar are verb phrases (**VP**s) and nominals (**N**s), this can be proved by an easy simulatenous induction on **N**s and **VP**s any by discarding all possible parse states where components do not satisfy co-indexing, polarity and voice constraints. For Y/N-questions we reason analogously.

($\Leftarrow$) We will prove, by induction on the body $\Phi[x]$ of a non-boolean GCQ q of distinguished variable x, that we can construct a question Q s.t. that q is the image of Q by $[\![.]\!]$. The result will then follow both for boolean and non-boolean GCQs. Recall that **N**s translate into unary predicates, **TV**s into binary predicates and **Pn**s into constants:

- **(Basis)** $q(x) \leftarrow \Phi[x]$ is the image of the question "which A_{i_0} who is a A_{i_1} who ... who is a A_{i_m} R_{j_0}s himself and ... and R_{j_m}s himself and R_{j_0}s c and ... and R_{j_m}s c and is R_{j_0}d by c and ... and is R_{j_m}d by c?".
- **(Inductive step)** $q(x) \leftarrow \Phi[x]$ is the image of the question "which $\Phi'[x]$ R_{j_0}s and R_{j_1}s and ... and R_{j_m}s some A_{i_0} who is a A_{i_1} and who is a A_{i_2} and ... and who is a A_{i_m} and who R_{j_0}s him and ... and who R_{j_m}s him and who $\Phi''[y]$?", by induction hypothesis on $\Phi'[x]$ and $\Phi''[y]$. □

Theorem 2. (Expressing QA) *The QA problem for Lite-English and GCQ-English falls under in* **LOGSPACE** *w.r.t. data complexity.*

Proof. It follows immediately from Theorem 1 and Lemma 1. □

5. Expressing Aggregate Queries

The question we now need to answer is: how can we expand the coverage of our CL without compromising the tractability of QA? In this section we propose to cover *graph-shaped aggregate queries*, that is, GCQs augmented with (some of) the basic SQL aggregation functions, COUNT, MIN, MAX and SUM. These functions are defined on finite subsets of $\mathbf{Dom} \cup \mathbb{Q}$, i.e., on DB domains plus the linearly ordered set of rational numbers and take values in $\mathbb{Q}$, that is, they compute a rational number. For the purposes of the current paper, we will restrict our analysis to only two of them, namely MAX and MIN, although this analysis can be easily generalized to cover all of these functions.

Aggregates arise frequently in domains and systems containing numerical data, e.g. geographical domains and systems. One of them, the GEOQUERY geography database system, comes with a NL interface that supports NL questions expressing such functions (Mooney 2007). The corpus of these questions showed that user questions did basically convey either a CQ or a CQ with aggregation functions (see Table 3). Most importantly,

	CQs	Aggregations	Negation
Questions	34.54%	65.35%	0.11%

Table 3: Frequency of CQs in GEOQUERY.

answering CQs (and a fortiori GCQs) with aggregation functions over DL-Lite ontologies is polynomial w.r.t. data complexity. So, how do these queries look like and what kind of questions do we want to have in our CL? We would like to capture queries over unary predicates computing a maximum like

(8) $$q(\mathbf{max}(n)) \leftarrow height(n) \wedge odd(n)$$

with a CL Wh-question like

(9) Which is the greatest height that is odd?

(Rule)	(Semantic action)
$\mathbf{VP}_{i,j} \rightarrow \mathbf{COP}\ \mathbf{NP}_j$	$[\![\mathbf{VP}_{i,j}]\!] := [\![\mathbf{COP}]\!]([\![\mathbf{NP}_j]\!])$
(Lexical rule)	(Value of $[\![.]\!]$ on word and category)
$\mathbf{Det} \rightarrow$ the greatest	$\lambda P.\mathbf{max}(P)\colon (\mathbb{Q} \rightarrow t) \rightarrow \mathbb{Q}$
$\mathbf{Det} \rightarrow$ the smallest	$\lambda P.\mathbf{min}(P)\colon (\mathbb{Q} \rightarrow t) \rightarrow \mathbb{Q}$
$\mathbf{Det} \rightarrow$ some	$\lambda P.\lambda Q.\exists n(P(n) \wedge Q(n))\colon (\mathbb{Q} \rightarrow t) \rightarrow ((\mathbb{Q} \rightarrow t) \rightarrow (\mathbb{Q} \rightarrow t))$
$\mathbf{Pro}_i \rightarrow$ something	$\lambda P.\exists n P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow t$
$\mathbf{Pro}_i^- \rightarrow$ anything	$\lambda P.\exists n P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow t$
$\mathbf{Pro}_i \rightarrow$ it	$\lambda P.P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow t$
$\mathbf{Pro}_i \rightarrow$ itself	$\lambda P.P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow t$
$\mathbf{Coord} \rightarrow$ and	$\lambda P.\lambda Q.\exists n(P(n) \wedge Q(n))\colon (\mathbb{Q} \rightarrow t) \rightarrow ((\mathbb{Q} \rightarrow t) \rightarrow (\mathbb{Q} \rightarrow t))$
$\mathbf{Relpro}_i \rightarrow$ that	$\lambda P.\lambda n.P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow (\mathbb{Q} \rightarrow t)$
$\mathbf{Intpro}_i \rightarrow$ which	$\lambda P.\lambda n.P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow (\mathbb{Q} \rightarrow t)$
$\mathbf{COP}_{i,j} \rightarrow$ is	$\lambda n.\lambda m.n \approx m\colon \mathbb{Q} \rightarrow (\mathbb{Q} \rightarrow t)$
$\mathbf{NP}_{gap_i} \rightarrow \epsilon$	$\lambda P.P(n)\colon (\mathbb{Q} \rightarrow t) \rightarrow t$
$\mathbf{N}_i \rightarrow$ height,...	$\lambda n.height(n)\colon \mathbb{Q} \rightarrow t$,...
$\mathbf{Adj} \rightarrow$ odd,...	$\lambda n.odd(n)\colon \mathbb{Q} \rightarrow t$,...

Table 4: Grammar rules for AGCQ-English.

Or queries computing a sum

(10) $$q(\mathbf{sum}(n)) \leftarrow height(n) \wedge odd(n)$$

with the question

(11) Which is the sum of all heights that are odd?

Definition 4. (AGCQs) A *graph-shaped conjunctive aggregate query* (AGCQ) over a relational schema **R** is a query of the form $q(\alpha(n)) \leftarrow \Phi[n]$, where $\alpha \in \{\mathbf{min}, \mathbf{max}\}$, n is q's distinguished variable, a numerical variable, and $\Phi[n]$ is the body of a non boolean GCQ. Note that there are no boolean AGCQs.

5.1. Expressing Aggregate Queries with AGCQ-English

To express AGCQs in CL we extend AGCQ-English into a new fragment of English called AGCQ-English as follows. Aggregation functions **min** and **max** are conveyed, in English, by, respectively, definite **NP**s like "the smallest **N**" and "the greatest **N**", only this time they must denote not a set of properties, but, instead, a numeric value. The symbol **N** stands for a nominal component that denotes sets of *numerical values*. The rest of the expression behaves in a manner similar to a determiner. We must thus start by enriching our set of primitive λ-FOL types from $\{e, t\}$ into $\{e, t, \mathbb{Q}\}$ and allow for new determiners of type $(\mathbb{Q} \rightarrow t) \rightarrow \mathbb{Q}$.

Definition 5. (Aggregate Determiners) An *aggregate determiner* is any of the following: **(i)** The determiner "the greatest", associated to **max** and of partial MR $\lambda P.\mathbf{max}(P)\colon (\mathbb{Q} \rightarrow t) \rightarrow \mathbb{Q}$. **(ii)** The determiner "the smallest", associated to the aggregation function **min** and of partial MR $\lambda P.\mathbf{min}(P)\colon (\mathbb{Q} \rightarrow t) \rightarrow \mathbb{Q}$.

Once aggregate determiners have been introduced, there are three steps left to finish covering aggregate queries with AGCQ-English. **(i)** We introduce a new interrogative

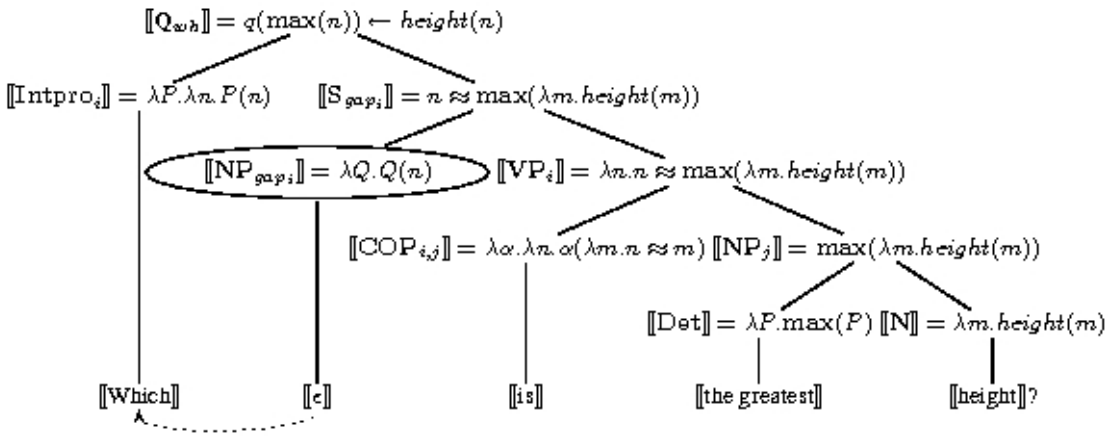

Figure 2: Translating "Which is the greatest height?".

pronoun "which" of semantics $\lambda P.\lambda n.(P)n\colon (\mathbb{Q} \to t) \to (\mathbb{Q} \to t)$, where P is a predicate symbol of type $\mathbb{Q} \to t$ (the type of sets of numbers) and n a variable of type $\mathbb{Q}$ (the type of numeric values). **(ii)** We introduce new entries for function words to take into account the new basic type $\mathbb{Q}$. **(iii)** We introduce the identity predicate "is" (of category **COP**, for copula) of semantics $\lambda n.\lambda m.m \approx n\colon \mathbb{Q} \to (\mathbb{Q} \to t)$. The reader can see in Table 4 the (new) lexical rules that extend CL coverage to aggregations. The semantic mapping $[\![.]\!]$ is then computed in the standard way over the parse tree of a AGCQ−English question, only it will now output, at the root of the tree a λ-FOL expression of the form $\lambda m.m \approx \alpha(\lambda n.\Phi[n])\colon \mathbb{Q} \to t$ that $[\![.]\!]$ will proceed to map into $q(\alpha(n)) \leftarrow \Phi[n]$. The reader can see a sample run of the procedure in Figure 2. Whence:

Lemma 2. *For every question Q in AGCQ−English, there exists a AGCQ q s.t. $[\![Q]\!] = q$. Conversely, every ATCQ q is the image by $[\![.]\!]$ of some question Q in AGCQ−English.*

Proof. (Sketch) As before, the first implication is proved by simultaneous induction on the **N**s and **TV**s of question Q. The second implication is proved by induction on the body of AGCQs q. □

Theorem 3. *QA is in* **P** *for Lite−English and AGCQ−English.*

Proof. It follows from Theorem 1 and Lemma 2. □

6. Conclusions and Further Work

We have provided a certain number of guidelines on how to characterize the computational complexity of CL interfaces to ontology-driven data access and management systems. This is achieved by expressing QA in controlled English. We have also shown that we reach tractability when we choose DL−Lite as ontology language and GCQs and AGCQs as query languages, for which two CLs, GCQ−English and AGCQ−English, have been introduced. As further work we plan to extend the coverage of AGCQ−English to the rest of the basic SQL functions, namely COUNT and SUM and to substantiate (or validate) the intuitiveness of these CLs anf of their English constructs by analysing more question corpora.

Acknowledgements

I would like to thank my supervisors, R. Bernardi and D. Calvanese, together with I. Pratt, for their help and suggestions.

References

R. Bernardi, et al. (2007). 'Lite Natural Language'. In *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*.

A. Bernstein, et al. (2003). 'Querying Ontologies: A Controlled English Interface for End-Users'. www.ifi.unizh.ch/ddis/staff/goehring.

D. Calvanese, et al. (2007). 'Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family'. *JAR* .

N. E. Fuchs, et al. (2005). 'Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces'. http://www.ifi.unizh.ch/attempto/publications.

D. Jurafsky & J. Martin (2000). *Speech and Language Processing*. Prentice Hall.

L. Lesmo & L. Robaldo (2007). 'Use of Ontologies in Practical NL Query Interpretation'. In *Proceedings of AI*IA 2007*.

M. Minock (2005). 'A Phrasal Approach to Natural Language Interfaces over Databases'. In *Natural Language Processing and Information Systems, 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*.

R. Montague (1970). 'Universal Grammar'. *Theoria* (36).

R. J. Mooney (2007). 'Learning for Semantic Parsing'. In *Proceedings of CICLing2007*.

I. Pratt (2001). 'On the Semantic Complexity of some Fragments of English'. Tech. rep., Department of Computer Science – University of Manchester.

R. Rosati (2007). 'The Limits of Querying Ontologies'. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*.

C. Thorne (2007). 'Managing Structured Data with Controlled English - An Approach Based on Description Logics'. In *Proceedings of ESSLLI 2007 Student Session*.

M. Vardi (1982). 'The Complexity of Relational Query Languages'. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*.