CAMILO THORNE
DIEGO CALVANESE

Tractability and Intractablity of Controlled Languages for Data Access

Abstract. In this paper we study the semantic data complexity of several controlled fragments of English designed for natural language front-ends to OWL (Web Ontology Language) and description logic ontology-based systems. Controlled languages are fragments of natural languages, obtained by restricting natural language syntax, vocabulary and semantics with the goal of eliminating ambiguity. Semantic complexity arises from the formal logic modelling of meaning in natural language and fragments thereof. It can be characterized as the computational complexity of the reasoning problems associated to their semantic representations. Data complexity (the complexity of answering a question over an ontology, stated in terms of the data items stored therein), in particular, provides a measure of the scalability of controlled languages to ontologies, since tractable data complexity implies scalability of data access. We present maximal tractable controlled languages and minimal intractable controlled languages.

Keywords: Controlled languages, semantic data complexity, controlled language interfaces to description logic ontologies, query answering.

1. Introduction

Controlled languages are ambiguity-free fragments of natural languages, obtained by simplifying, modifying and constraining the vocabulary, syntax and semantics of natural languages so that their utterances give rise to a unique parsing and semantic interpretation. Such semantic interpretation is obtained compositionally and can map controlled language utterances into logic expressions using the machinery of Montague semantics [13].

The tight integration with logic of controlled languages has lead to the proposal of controlled languages and of controlled language interfaces to ontology-based systems centered around the W3C ontology language standard, OWL * (Web Ontology Language) [16, 8, 11], which is formally underpinned by description logics [3, 9], a family of knowledge representation formalisms based on decidable fragments of first order logic (FO). One such example is ACE-OWL, which maps into OWL DL [11, 8].

OWL ontologies give a global, unified view of the the system's domain of interest in terms of a high level conceptualization that describes the basic concepts, relations, and constraints that hold in such a domain. The data itself can be stored in different formats but is typically structured and stored in relational databases

*`http://www.w3.org/TR/owl-ref/`

or triple stores [17]. Controlled language interfaces make ontology-based systems more usable for casual users. In this context, two tasks are targeted: *(i)* authoring ontologies (i.e., declaring and updating domain constraints, or declaring and updating factual information about the domain) *(ii)* accessing information stored in ontologies (i.e., evaluating information requests). This means that controlled languages need to map not only into ontology languages such as OWL, but also into the (formal) query languages used to access information [16, 8, 11].

An important issue that arises is the computational cost of semantically processing and reasoning with controlled languages, also known as their *semantic complexity*. Semantic complexity is particularly relevant for ontology-based system front-ends, given the nature of the data management tasks targeted. Moreover, as data in ontology-based systems is usually very large, it is important to focus on the so-called *data complexity* of semantic processing and reasoning. Data complexity has been formally characterized as the computational complexity of managing ontologies measured w.r.t. the size of the dataset [20, 6], and provides a measure of scalability of inference and data management tasks in ontology-based systems based on OWL [17].

Data complexity depends on the different combinations of constructs supported by the controlled language and the controlled language interface (the controlled language's function and content words). Hence, different controlled language design choices will have a positive or negative impact on this measure. The data complexity of accessing ACE-OWL Lite (the fragment of ACE-OWL that maps to OWL Lite) is CONP-complete. This is because the data complexity of query answering in the description logic that underpins OWL Lite, $\mathcal{SHIF}[D]$, is known to be data complete for CONP [14]. Hence, controlled languages like ACE-OWL do not scale to data, but might contain fragments that do. This raises the issue of determining which English constructs the tractable fragments cover.

This paper makes three contributions. Firstly, we define several declarative controlled fragments of English that express distinct fragments of OWL Lite and $\mathcal{SHIF}[D]$ (and can be seen as fragments of ACE-OWL). Secondly, we define an interrogative controlled language that expresses a significant number of so-called conjunctive queries, viz., SELECT-PROJECT-JOIN SQL queries, one of the standard query languages for OWL. Thirdly, we study the data complexity of accessing data via those controlled (declarative and interrogative) languages and show which *maximal* combinations of controlled language constructs allow for tractable data complexity and which *minimal* combinations give rise to intractable data complexity.

2. Controlled Languages and Semantic Complexity

The semantics, semantic processing and semantic complexity of controlled languages can be modelled using the standard formal and computational Montagovian analysis for English as developed in [4, 13]. Formal semantics provides a logic-based account of natural language semantics and semantic compositionality by assigning *meaning representations* to controlled language constituents, viz., logical expressions that model meaning. The logic from which these logical expressions are taken is λ-FO, viz., FO enriched with the constructs of the simply-typed lambda calculus. While this takes us beyond FO, the key issue is that λ-FO constructs serve only to "glue" together semantic representations compositionally, and to obtain FO semantic semantic representations for full utterances. Controlled languages and fragments can be defined using, essentially, context-free grammars with semantic actions of compiler theory (see [10], Chapter 18) or Montague grammars (see [13]). Semantic actions ensure that the controlled fragment maps into a logic by defining a *compositional translation* $\tau(\cdot)$.

Meaning Representations. Let $\{v_i \mid i \in \mathbb{N}\}$ be a set of λ-FO *variables*, and $\{k_i \mid i \in \mathbb{N}\}$ a set of λ-FO *constants*. Let t stand for the type of Boolean values and e for the type of individual constants. The set of λ-FO *expressions* u and *types* T are defined, resp., by the grammars $u ::= k_i \mid v_i \mid \lambda v_i.u \mid u(u')$ and $T ::= e \mid t \mid T{\rightarrow}T'$.

A *typing* is a function $\chi(\cdot)$ from λ-FO expressions into types. If $\chi(u) = T$ we write $u{:}T$. The *typing rule* is the following application rule:

$$\textit{app}\ \frac{u{:}T \qquad u'{:}T{\rightarrow}T'}{u'(u){:}T'}$$

We say that a term $\lambda v_i.u(u')$ *reduces* to an expression u'', whenever u'' is the result of deleting the prefix λv_i and susbtituting, consistently with typings and typing rules, each occurrence of v_i in u by u'. The *beta reduction* $\rhd$ relation, is the reflexive and transitive closure of the reduction relation.

Note that λ-FO variables, constants and expressions are used in a more general way than their FO or description logic counterparts and may denote arbitrary objects of arbitrary complexity. A variable or constant of type e will denote an individual (a simple object, such as an integer), whereas a variable or constant of type $e{\rightarrow}t$, will denote a characteristic function (or set, a complex object).

In what follows we will avoid explicit typing and adopt instead the following conventions. Expressions of type e (denoting individuals) will be written with lower case letters (x, y, z, etc.), expressions of type $e \times \cdots \times e{\rightarrow}t$ (denoting sets and relations) will be written with upper case letters (P, Q, R, etc.) and expressions

of type $(e{\rightarrow}t){\rightarrow}t$ (denoting so-called generalized quantifiers) with Greek letters (α, β, γ, etc.).

Controlled Fragments. *Semantically enriched grammars* are context-free grammar G built from *(i)* a set $\mathbf{Sig}$ of *words*, *(ii)* a set $\mathbf{Cat}$ of syntactic *categories*, *(iii)* a *lexicon* $\mathbf{Lex} \subseteq \mathbf{Cat} \times \mathbf{Sig}$, *(iv)* a set $\mathbf{Rul} \subseteq \mathbf{Cat} \times (\mathbf{Sig} \cup \mathbf{Cat})^+$ of *phrase structure rules*, *(v)* a distinguished category $\mathbf{S}$ (or $\mathbf{Q}$) called the *start* category, and *(vi)* a *compositional translation* $\tau(\cdot)$. If $(C, w) \in \mathbf{Lex}$ or $(C, w) \in \mathbf{Rul}$ we write $C ::= w$. Any sequence $w \in (\mathbf{Sig} \times \mathbf{Cat})^*$ is called a syntactic *constituent*.

The notions of *derivation in one step*, denoted $\Longrightarrow$, of derivation, denoted $\Longrightarrow^*$ (the reflexive and transitive closure of $\Longrightarrow$), and of *parse tree* are defined as for standard context-free grammars (see [10], Chapter 18). By $\Longrightarrow^k$ we denote derivations of $\leq k$ steps.

Compositional translations $\tau(\cdot)$ map syntactic constituents to λ-FO meaning representations. By exploiting the phrase structure rules and lexicon of a grammar G, $\tau(\cdot)$ can be recursively defined on syntactic constituents by means of *semantic actions*: *(i)* for each $C ::= w \in \mathbf{Lex}$, we specify $\tau(C)$, and *(ii)* for each $C ::= C_1 \cdots C_n \in \mathbf{Rul}$, we write $\tau(C) := \tau(C_{\pi(1)})(\ldots\tau(C_{\pi(n)})\ldots)$, where $\pi(\cdot)$ is a permutation (necessary to allow for different word orders in the target and source languages). Complete sentences translate into FO sentences (closed formulas without free variables).

Meaning representations induce a partitioning of the lexicon into a set of *content words*, wherein words (nouns and verbs) denote sets, relations, or individuals, and a set of *function words*, in which words (determiners, pronouns, articles, coordinating particles, etc.) denote operations over such sets, relations, and individuals.

Parsing and semantic interpretation occur as follows. Firstly, a parse tree is computed. Secondly, semantic actions are applied bottom-up, from leaves to root. Thirdly, applications and beta reductions are effectuated, in observance with typing rules. If parsing and semantic interpretation succed, utterances are considered grammatical, otherwise non-grammatical. Accordingly, the language *generated* by grammar G is defined as $L(G) := \{w \in \mathbf{Sig}^* \mid \mathbf{S} \Longrightarrow^* w \text{ and } \tau(w) \text{ is defined}\}$.

Semantic Complexity. Controlled languages and fragments give rise to *logic fragments* with specific computational properties. Given a fragment L (defined by a grammar G), the logic fragment *expressed* by L is the FO fragment $\tau(L) := \{\tau(w) \mid w \in L\}$. Following Pratt in [15], we define the *semantic complexity* of a (controlled) fragment of English L as the computational complexity of reasoning with the formulas belonging to $\tau(L)$.

$$
\begin{array}{rlrl}
R^{t_x} &:= R(x,y) & R^{t_y} &:= R(y,x) \\
(R^-)^{t_x} &:= R(y,x) & (R^-)^{t_y} &:= R(x,y) \\
A^{t_x} &:= A(x) & A^{t_y} &:= A(y) \\
(\exists S)^{t_x} &:= \exists y(S^{t_x}) & (\exists S)^{t_y} &:= \exists x(S^{t_y}) \\
(\neg A_f)^{t_x} &:= \neg A_f^{t_x} & (\neg A_f)^{t_y} &:= \neg A_f^{t_y} \\
(\neg\exists S)^{t_x} &:= \neg\exists y(S^{t_x}) & (\neg\exists S)^{t_y} &:= \neg\exists x(S^{t_y}) \\
(\exists S{:}C_f)^{t_x} &:= \exists y(S^{t_x} \wedge C_f^{t_y}) & (\exists S{:}C_f)^{t_y} &:= \exists x(S^{t_y} \wedge C_f^{t_x}) \\
(C_f \sqcap C'_f)^{t_x} &:= C_f^{t_x} \wedge C'^{t_x}_f & (C_f \sqcap C'_f)^{t_y} &:= C_f^{t_y} \wedge C'^{t_y}_f \\
(C_f \sqcup C'_f)^{t_x} &:= C_f^{t_x} \vee C'^{t_x}_f & (C_f \sqcup C'_f)^{t_y} &:= C_f^{t_y} \vee C'^{t_y}_f \\
(\forall S{:}C_f)^{t_x} &:= \forall y(S^{t_x} \Rightarrow C_f^{t_y}) & (\forall S{:}C_f)^{t_y} &:= \forall x(S^{t_y} \Rightarrow C_f^{t_x}) \\
(C_l \sqsubseteq C_r)^{t_x} &:= \forall x(C_l^{t_x} \Rightarrow C_r^{t_x}) & (C_l \sqsubseteq C_r)^{t_y} &:= \forall y(C_l^{t_y} \Rightarrow C_r^{t_y})
\end{array}
$$

Figure 1. The canonical $\cdot^{t_x}$ and $\cdot^{t_y}$ translations (where $f \in \{l, r\}$).

3. Ontology Languages and Semantic Complexity

OWL-based systems are underpinned by description logics [6, 3], which are fragments of FO written in an object-oriented syntax. They structure the domain of discourse in terms of concepts (representing classes, i.e., sets of objects) and roles (representing binary relations between objects). Data is accessed by means of formal queries, based on the SQL (Structured Query Language) standard for relational databases. Since ontology-based systems may store and manage large amounts of data, determining the *data complexity* of data access, i.e., the complexity measured in terms of the size of the data only, is a key issue. In this paper we will focus on description logics defined by restricting the syntax of the logic $\mathcal{ALCI}$ and hence of OWL DL and OWL Lite (which correspond to a superlanguage of $\mathcal{ALCI}$) [3].

Description Logic Ontologies and Knowledge Bases. In an *$\mathcal{ALCI}$ ontology* $\mathcal{O}$, intensional knowledge is specified by means of a set of (concept inclusion) *assertions* σ of the form $C_l \sqsubseteq C_r$, stating inclusion (or IS-A) between the instances of the *left concept* C_l and those of the *right concept* C_r. Consider a countable signature of *concept names* A (unary predicates) and *role names* R (binary predicates). A *role* S is either a role name R or its *inverse* R^-. A left or right concept is either a concept name A, an unqualified existential restriction $\exists R$, the negation $\neg A$, $\neg\exists R$ of A or $\exists R$, or, if C_f and C'_f are concepts, for $f \in \{l, r\}$, *(i)* a qualified existential restriction $\exists S{:}C_f$, *(ii)* an intersection $C_f \sqcap C'_f$, *(iii)* a union $C_f \sqcup C'_f$, or *(iv)* a qualified universal restriction $\forall S{:}C_f$.

Different combinations of left and right concepts give rise to different frag-

ments of $\mathcal{ALCI}$ and hence, to different description logics. An important description logic defined in this manner is *DL-Lite*$_\sqcap$ [6, 2], of syntax

$$S ::= R \mid R^-,\ \ C_l ::= A \mid \exists S \mid C_l \sqcap C_l',\ \ C_r ::= \neg A \mid \neg\exists S \mid C_l \mid C_r \sqcap C_r',$$

which is contained in the Horn fragment of the two-variable fragment of FO. It is important because it can capture the fundamental features of conceptual modelling formalisms (UML class diagrams, ER-diagrams, etc.) and formally underpins efficient ontology-based systems [6, 2].

A *database*, expressing extensional knowledge, is a finite set $\mathcal{D}$ of FO unary and binary ground atoms (a.k.a. facts) of the form $A(c)$, $R(c, c')$, where A is a concept name, R a role, name and c and c' constants. A *knowledge base* is a pair $(\mathcal{O}, \mathcal{D})$, where $\mathcal{O}$ is an ontology and $\mathcal{D}$ a database.

The semantics of description logic concepts and roles can be specified by means of a pair of so-called canonical translations $\cdot^{t_x}$ and $\cdot^{t_y}$ into FO (specifically, to the two-variable guarded fragment of FO), as shown in Figure 1.

We consider FO semantics under the so-called *standard names assumption*. That is, we consider a fixed countably infinite domain $\mathbf{Dom} := \{c_i \mid i \in \mathbb{N}\}$ of constants (interpreted by themselves) and define *(FO) interpretations* as tuples $\mathcal{I} := (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ where $\mathbb{D}_\mathcal{I} \subseteq \mathbf{Dom}$ and $\cdot^\mathcal{I}$ is an interpretation function. Thereafter, the notions of entailment, truth, and satisfaction are defined as usual. In particular, we say that an interpretation $\mathcal{I}$ is a *model* of a formula ψ, written $\mathcal{I} \models \psi$, if ψ evaluates to true under some satisfying assignment over $\mathcal{I}$. For every set Γ of formulas, we write $\mathcal{I} \models \Gamma$ if $\mathcal{I} \models \psi$ for all $\psi \in \Gamma$. We say that ψ' (resp., Γ) *entails* ψ, written $\psi' \models \psi$ (resp., $\Gamma \models \psi$), if every model of ψ' (resp., Γ) is a model of ψ. When also the converse holds, we say that ψ and ψ' are *equivalent*.

An interpretation $\mathcal{I}$ is said to be a *model* of an inclusion assertion $\sigma = C_l \sqsubseteq C_r$, in symbols $\mathcal{I} \models \sigma$, if $\mathcal{I} \models (C_l \sqsubseteq C_r)^{t_x}$. It is said to be a *model* of a knowledge base $(\mathcal{O}, \mathcal{D})$, in symbols $\mathcal{I} \models (\mathcal{O}, \mathcal{D})$, if it is a model of every assertion in $\mathcal{O}$ and every fact $P(c)$, $R(c, c')$ in $\mathcal{D}$. Notice that we interpret databases under the *open world assumption*, and thus facts not in $\mathcal{D}$ may or may not hold in the model $\mathcal{I}$.

Formal Queries. We use queries to retrieve information from knowledge bases. In the setting of ontology languages it is customary to consider fragments of SQL. In this paper we will consider two such fragments.

A *conjunctive query* (CQ) φ is a (positive existential) FO formula of the form $\exists \bar{y}(\varphi'(\bar{x}, \bar{y}))$, where $\bar{x}$ denotes a (finite) sequence of variables of *length* $|\bar{x}|$ (the free variables of φ), called the query's *distinguished variables*, and $\varphi'(\bar{x}, \bar{y})$ is a conjunction of FO relational atoms over the variables $\bar{x}$ and $\bar{y}$. If $\bar{x}$ is the empty sequence, we say that the query is *boolean*. CQs constitute a FO specification of an SQL SELECT-PROJECT-JOIN query [1].

Graph-shaped conjunctive queries (GCQs) are a restricted kind of CQs with at most one distinguished variable x, which are inductively defined by[†]

$$\begin{aligned} \varphi(x) \ ::= \ & A(x) \mid R(x,x) \mid R(x,c) \mid \exists y(R(x,y)) \mid \exists y\,(S(x,y) \wedge \varphi'(y)) \mid \\ & \varphi'(x) \wedge \varphi''(x), \\ S(x,y) \ ::= \ & R(x,y) \mid R(y,x) \mid S'(x,y) \wedge S''(x,y) \end{aligned}$$

GCQs are a linguistically motivated class of CQs, easily expressible by English sentence subordination. Moreover, they allow (as we shall see later) to consider some interesting phenomena, such as anaphora. See also [18], Ch. 4.

Let φ be a (G)CQ. A *grounding* is a (not necessarily total) function $\theta(\cdot)$ that maps the variables of φ to $\mathbf{Dom}$. Groundings are extended to complex syntactic objects in the usual way. We denote by $\varphi\theta$ the *grounding* of φ by $\theta(\cdot)$. Queries are FO formulas. Consequently, following FO semantics, an interpretation $\mathcal{I}$ is a *model* of $\varphi\theta$, if $\varphi\theta$ evaluates to true in $\mathcal{I}$ under some variable assignment.

We say that a sequence $\bar{c}$ of constants is an *answer* to (G)CQ φ with distinguished variables $\bar{x}$ over a knowledge base $(\mathcal{O}, \mathcal{D})$, if there exists a grounding $\theta(\cdot)$, mapping $\bar{x}$ to $\bar{c}$, such that $\varphi\theta$ is *logically entailed* by $(\mathcal{O}, \mathcal{D})$, viz., whenever, for all $\mathcal{I}$, $\mathcal{I} \models (\mathcal{O}, \mathcal{D})$ implies $\mathcal{I} \models \varphi\theta$; in symbols: $(\mathcal{O}, \mathcal{D}) \models \varphi\theta$. We illustrate these notions with a simple example.

EXAMPLE 3.1. Consider an ontology about people, defined by $\mathcal{O}_p := \{\textit{Man} \sqsubseteq \textit{Person},\ \textit{Woman} \sqsubseteq \textit{Person},\ \textit{Man} \sqsubseteq \exists \textit{Loves},\ \exists \textit{Loves} \sqsubseteq \exists \textit{Likes}\}$, which states that men and women are persons, that men love somebody, and that who loves also likes. Consider now the database $\mathcal{D}_p := \{\textit{Man}(\text{john}), \textit{Woman}(\text{mary})\}$ stating that John is a man and that Mary is a woman. Suppose we want to know which man likes somebody. This information request can be captured by the GCQ $\varphi_p := \exists y\,(\textit{Man}(x) \wedge \textit{Likes}(x,y))$. Clearly, John is the only answer: the only grounding satisfying the entailment condition is $\theta_p := \{x \mapsto \text{john}\}$. ♣

Semantic Data Complexity. As we said earlier, the semantic complexity of a controlled language L consists in the computational complexity of the logical reasoning problems for the logic $\tau(L)$ it expresses. For controlled languages designed for controlled language front-ends to ontology-based systems, and which, therefore, express formal ontology and query languages, this means focusing on the complexity of one key problem: accessing information.

[†] A CQ can be mapped into a labelled graph, with each variable or constant giving rise to a vertex, each unary atom $A(x)$ to a label A of vertex x, and each binary symbol $R(x,y)$ to a directed edge from x to y with label R. For a GCQ, such a graph has the structure of a tree (possibly with multiple labels on the same edge).

The *query answering* problem for (G)CQs and knowledge bases (KBQA) is the reasoning (entailment) problem stated as follows: **Input:** a knowledge base $(\mathcal{O}, \mathcal{D})$, a (G)CQ φ with distinguished variables $\bar{x}$, and a sequence $\bar{c}$ of $|\bar{x}|$ constants, **Question:** does there exist a grounding $\theta(\cdot)$ s.t. $\theta(\bar{x}) = \bar{c}$ and $(\mathcal{O}, \mathcal{D}) \models \varphi\theta$?

In addition, we are interested in the *data complexity* of KBQA, namely, in its computational complexity when we consider $\mathcal{D}$ as the only input of the problem [20]. This is because in real-world ontology-based systems data can be very large, and in particular much larger than the intensional specification $\mathcal{O}$. Thus, one can abstract away from ontologies and queries and assume that tractable (i.e., PTIME) data complexity for KBQA provides an upper bound for the system's scalability to large data repositories.

Notice moreover, that all the other data management or reasoning tasks and problems relevant to ontologies and ontology-based systems, reduce to KBQA, and thus KBQA provides (data) complexity upper bounds for all of them (see, e.g., [6]).

4. Expressing Ontology Languages

In this section we describe a methodology for defining controlled languages that express *all* the possible combinations of left and right concepts and concept descriptions discussed in the preceding section. The main ideas are three. *(i)* We consider grammar rules that express each such concept. *(ii)* Syntactic categories and constituents are *subcategorized* (or multiplied) into *left* and *right* categories and constituents (by means of the indexes or features l and r, respectively), to capture the distinction between left and right concepts. *(iii)* By combining such rules we express the concept combinations. We will see that this is essential to determine *maximally* tractable and *minimally* intractable controlled fragments w.r.t. data complexity.

We consider as content words common nouns (non-recursive $\mathbf{N}$s), transitive and intransitive verbs ($\mathbf{TV}$s and $\mathbf{IV}$s) and adjectives ($\mathbf{Adj}$s). Words from these categories denote (atomic) concept names and (atomic) role names. Content words are glued together into complete sentences by function words: determiners ($\mathbf{Det}$s), (indeterminate) pronouns ($\mathbf{Pro}$s), relative pronouns ($\mathbf{Relp}$s), conjunctions, disjunctions ($\mathbf{Crd}$s) and negations ($\mathbf{Neg}$s). Regarding non-lexical categories, we consider verb phrases ($\mathbf{VP}$s), noun phrases ($\mathbf{NP}$s), nominals (recursive $\mathbf{N}$s [‡]), and complete sentences ($\mathbf{S}$s). In particular, recursive (i.e., nominals) and non-recursive (i.e., nouns) $\mathbf{N}$s and $\mathbf{VP}$s map into arbitrary description logic concepts, or, more precisely, into set-typed expressions (of type $e{\rightarrow}t$). Finally, our fragments allow for

[‡]We follow the tradition in using the same category for both nominals and nouns.

Table 1. Expressing concepts C_f with constituents w_{C_f}, with $f, f' \in \{l, r\}$. We omit content lexicon entries. The rules are divided into grammar rules and function lexicon entries (if any). Notice the use of non-standard meaning representations for "somebody" and "only".

C_f	w_{C_f}	**Rules and Semantic Actions** G_{C_f}	
A	$\mathbf{N}_f$, $\mathbf{VP}_f$.	$\mathbf{VP}_f ::=$ is a $\mathbf{N}_f$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{N}_f)$
		$\mathbf{VP}_f ::= \mathbf{IV}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{IV})$
		$\mathbf{VP}_f ::=$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{Adj})$
		$\mathbf{N}_f ::= \mathbf{N}$	$\tau(\mathbf{N}_f) := \tau(\mathbf{N})$
$\exists R$	$\mathbf{TV}$ something.	$\mathbf{VP}_f ::= \mathbf{TV}\,\mathbf{NP}_f$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_f))$
		$\mathbf{NP}_f ::= \mathbf{Pro}_f$	$\tau(\mathbf{NP}_f) := \tau(\mathbf{Pro}_f)$
		$\mathbf{Pro}_f ::=$ something	$\tau(\mathbf{Det}_f) := \lambda P.\exists x P(x)$
$\neg A$, $\neg(\exists R)$	is not $\mathbf{Adj}$, does not $\mathbf{TV}$, is not a $\mathbf{N}_{f'}$.	$\mathbf{VP}_f ::=$ does $\mathbf{Neg}\,\mathbf{TV}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{Neg})(\tau(\mathbf{TV}))$
		$\mathbf{VP}_f ::=$ is $\mathbf{Neg}\,\mathbf{Adj}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{Neg})(\tau(\mathbf{Adj}))$
		$\mathbf{VP}_f ::=$ is $\mathbf{Neg}$ a $\mathbf{N}_{f'}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{Neg})(\tau(\mathbf{N}_{f'}))$
		$\mathbf{Neg} ::=$ not	$\tau(\mathbf{Neg}) := \lambda P.\lambda x.\neg P(x)$
$\exists R{:}C_{f'}$	$\mathbf{TV}$ some $\mathbf{N}_{f'}$, $\mathbf{TV}$ somebody who $\mathbf{VP}_{f'}$.	$\mathbf{VP}_f ::= \mathbf{TV}\,\mathbf{NP}_{f'}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_{f'}))$
		$\mathbf{VP}_f ::= \mathbf{TV}\,\mathbf{Pro}_{f'}\,\mathbf{Relp}_{f'}\,\mathbf{VP}_{f'}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{TV})(\tau(\mathbf{Pro}_{f'})(\tau(\mathbf{Relp}_{f'})(\tau(\mathbf{VP}_{f'}))))$
		$\mathbf{NP}_f ::= \mathbf{Det}_{f'}\,\mathbf{N}_{f'}$	$\tau(\mathbf{NP}_f) := \tau(\mathbf{Det}_{f'})(\tau(\mathbf{N}_{f'}))$
		$\mathbf{Det}_{f'} ::=$ some	$\tau(\mathbf{Det}_{f'}) := \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
		$\mathbf{Pro}_{f'} ::=$ somebody	$\tau(\mathbf{Det}_{f'}) := \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
		$\mathbf{Relp}_{f'} ::=$ who	$\tau(\mathbf{Det}_{f'}) := \lambda P.\lambda x.P(x)$

C_f	w_{C_f}	**Rules and Semantic Actions** G_{C_f}	
$C_{f'} \sqcap C'_{f'}$	$\mathbf{Adj}\,\mathbf{N}_{f'}$, $\mathbf{N}_{f'}$ who $\mathbf{VP}_{f'}$, $\mathbf{N}_{f'}$ and $\mathbf{N}_{f'}$, $\mathbf{VP}_{f'}$ and $\mathbf{VP}_{f'}$.	$\mathbf{VP}_f ::= \mathbf{VP}_{f'}\,\mathbf{Crd}\,\mathbf{VP}_{f'}$	$\tau(\mathbf{VP}_f) := (\tau(\mathbf{Crd})(\tau(\mathbf{VP}_{f'})))(\tau(\mathbf{VP}_{f'}))$
		$\mathbf{N}_f ::= \mathbf{Adj}\,\mathbf{N}_{f'}$	$\tau(\mathbf{N}_f) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_{f'}))$
		$\mathbf{N}_f ::= \mathbf{N}_{f'}\,\mathbf{Crd}\,\mathbf{N}_{f'}$	$\tau(\mathbf{N}_f) := (\tau(\mathbf{Crd})(\tau(\mathbf{N}_{f'})))(\tau(\mathbf{N}_{f'}))$
		$\mathbf{N}_f ::= \mathbf{N}_{f'}\,\mathbf{Relp}_{f'}\,\mathbf{VP}_{f'}$	$\tau(\mathbf{N}_f) := (\tau(\mathbf{Relp}_{f'})(\tau(\mathbf{N}_{f'})))(\tau(\mathbf{VP}_{f'}))$
		$\mathbf{Relp}_{f'} ::=$ who	$\tau(\mathbf{Relp}_{f'}) := \lambda P.\lambda Q.\lambda x.(Q(x) \wedge P(x))$
		$\mathbf{Relp}_{f'} ::=$ that	$\tau(\mathbf{Relp}_{f'}) := \lambda P.\lambda Q.\lambda x.(Q(x) \wedge P(x))$
		$\mathbf{Crd} ::=$ and	$\tau(\mathbf{Crd}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$C_{f'} \sqcup C'_{f'}$	$\mathbf{N}_{f'}$ or $\mathbf{N}_{f'}$, $\mathbf{VP}_{f'}$ or $\mathbf{VP}_{f'}$.	$\mathbf{VP}_f ::= \mathbf{VP}_{f'}\,\mathbf{Crd}\,\mathbf{VP}_{f'}$	$\tau(\mathbf{VP}_f) := (\tau(\mathbf{Crd})(\tau(\mathbf{VP}_{f'})))(\tau(\mathbf{VP}_{f'}))$
		$\mathbf{N}_f ::= \mathbf{N}_{f'}\,\mathbf{Crd}\,\mathbf{N}_{f'}$	$\tau(\mathbf{N}_f) := (\tau(\mathbf{Crd})(\tau(\mathbf{N}_{f'})))(\tau(\mathbf{N}_{f'}))$
		$\mathbf{Crd} ::=$ or	$\tau(\mathbf{Crd}) := \lambda P.\lambda Q.\lambda x.(P(x) \vee Q(x))$
$\forall R{:}C_{f'}$	$\mathbf{TV}$ only $\mathbf{N}_{f'}$, $\mathbf{TV}$ only who $\mathbf{VP}_{f'}$.	$\mathbf{VP}_f ::= \mathbf{TV}\,\mathbf{NP}_{f'}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_{f'}))$
		$\mathbf{VP}_f ::= \mathbf{TV}\,\mathbf{Pro}_{f'}\,\mathbf{Relp}_{f'}\,\mathbf{VP}_{f'}$	$\tau(\mathbf{VP}_f) := \tau(\mathbf{TV})(\tau(\mathbf{Pro}_{f'})(\tau(\mathbf{Relp}_{f'})(\tau(\mathbf{VP}_{f'}))))$
		$\mathbf{NP}_f ::= \mathbf{Det}_{f'}\,\mathbf{N}_{f'}$	$\tau(\mathbf{NP}_f) := \tau(\mathbf{Det}_{f'})(\tau(\mathbf{N}_{f'}))$
		$\mathbf{Det}_{f'} ::=$ only	$\tau(\mathbf{Det}_{f'}) := \lambda P.\lambda Q.\forall x(Q(x) \Rightarrow P(x))$
		$\mathbf{Pro}_{f'} ::=$ only	$\tau(\mathbf{Pro}_{f'}) := \lambda P.\lambda Q.\forall x(Q(x) \Rightarrow P(x))$
		$\mathbf{Relp}_{f'} ::=$ who	$\tau(\mathbf{Relp}_{f'}) := \lambda P.\lambda x.P(x)$

Table 2. Grammar rules and function lexicon capturing IS-A. Notice the use of non-standard meaning representations for "anybody" and "anything".

Rules and Semantic Actions $G_{\sqsubseteq}$	
$\mathbf{S} ::= \mathbf{NP}_l\, \mathbf{VP}_r$	$\tau(\mathbf{S}_l) := \tau(\mathbf{NP}_l)(\tau(\mathbf{VP}_r))$
$\mathbf{NP}_l ::= \mathbf{Pro}_l\, \mathbf{Relp}_l\, \mathbf{VP}_l$	$\tau(\mathbf{NP}_l) := \tau(\mathbf{Pro}_l)(\tau(\mathbf{Relp}_l)(\tau(\mathbf{VP}_l)))$
$\mathbf{NP}_l ::= \mathbf{Det}_l \mathbf{N}_l$	$\tau(\mathbf{NP}_l) := \tau(\mathbf{Det}_l)(\tau(\mathbf{N}_l))$
$\mathbf{Pro}_l ::=$ anybody	$\tau(\mathbf{Pro}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x))$
$\mathbf{Relp}_l ::=$ who	$\tau(\mathbf{Relp}_l) := \lambda P.\lambda x.P(x)$
$\mathbf{Pro}_l ::=$ anything	$\tau(\mathbf{Pro}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x))$
$\mathbf{Relp}_l ::=$ that	$\tau(\mathbf{Relp}_l) := \lambda P.\lambda x.P(x)$
$\mathbf{Det}_l ::=$ every	$\tau(\mathbf{Det}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x))$

sentence subordination; subordinate clauses are captured, basically, by combining $\mathbf{VP}$s with $\mathbf{Relp}$s and $\mathbf{N}$s. See Table 1.

When designing controlled declarative languages that map into description logics, all such languages will share the following common features. Sentences map to IS-A assertions of the form $C_l \sqsubseteq C_r$. Therefore, all utterances will comply with the sentence patterns:

$$\text{"every } w_{C_l}\ w_{C_r}\text{"} \qquad \text{and} \qquad \text{"everybody who } w_{C_l}\ w_{C_r}\text{"}.$$

The controlled languages are defined around the core grammar rules $G_{\sqsubseteq}$ from Table 2, which are then combined with the subset of the rules G_{C_f} from Table 1, defining the constituents w_{C_f}, for $f \in \{l, r\}$, that correspond to the description logic constructs allowed respectively in the left and right hand side of inclusion assertions. We impose that all such fragments must express IS-A among atomic concepts, i.e., they will all contain also the rules $G_{\sqsubseteq} \cup G_{A_l} \cup G_{A_r}$. Notice that we do not consider passives, and hence do not directly express (qualified) inverted roles. However, this can be easily accomplished by adding rules like:

$$\mathbf{VP}^p_f ::= \text{ is } \mathbf{TV} \text{ by } \mathbf{NP}_f \qquad \tau(\mathbf{VP}^p_f) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_f)),$$

to the rules $G_{(\exists R)_f}$, $G_{\neg(\exists R)_f}$, $G_{(\exists R:C_{f'})_f}$ and $G_{(\forall R:C_{f'})_f}$, for $f, f' \in \{l, r\}$. This gives as a result a large number of possible fragments, even if we disregard passives, morphosyntactic features (number, gender, tense, person, polarity, etc.) and agreement.

Notice that, while a certain degree of lexical ambiguity may remain, we will obtain fragments whose complete utterances map into a unique meaning represen-

tation (a unique assertion $C_l \sqsubseteq C_r$). To ensure this, as well as to ensure that complete utterances correctly express assertions, non-standard meaning representations are sometimes associated to English (function) words and constituents (e.g., the $\mathbf{Pro}$ "anybody" in Table 2 has been assigned the same semantics and typing of the $\mathbf{Det}$ "every", see also [5]).

At each state of parsing (which can be seen as a walk through a tree-shaped space of so-called partial parse trees or parsing states), a constituent w_{C_f} of index (or feature) f, for $f \in \{l, r\}$, of meaning representation C_f, and of type T will be generated. The typing rules and features will prune undesired parse trees (or parse states), thereby yielding only the left or right concepts the grammar rules from Table 1 express.

5. Expressing *DL-Lite*$_\sqcap$ and GCQs

To study the semantic data complexity of controlled languages for ontology-based data access three requirements must be fulfilled. *(i)* We need to consider the different combinations of English content and function words and the description logics they give rise to. *(ii)* We need to consider a controlled language that expresses GCQs. *(iii)* We need to derive data complexity results for answering such controlled questions over the ontologies the declarative languages defined by Table 1 express. The results in this section generalize results from [5] and [18], Ch. 4.

Lite-English. Controlled languages for which KBQA is in AC^0 in data complexity can be considered optimal for data access over ontologies and ontology-based data access systems, since this complexity matches the one of query evaluation in plain relational databases. One such declarative controlled language is Lite-English, which expresses *DL-Lite*$_\sqcap$. Lite-English is defined by considering the following combination of grammar rules from Table 1: $G_\sqsubseteq \cup G_{(\neg A)_r} \cup G_{(\neg \exists R)_r} \cup G_{A_f} \cup G_{(\exists R)_f} \cup G_{(C_f \sqcap C_f)_f}$, for $f \in \{l, r\}$.

To simplify the proofs that follow, we introduce the notion of *structural equivalence* and *equivalence*. A λ-FO expression $u = \lambda v_{i_1} \ldots \lambda v_{i_n}.u'$ is said to be *structurally equivalent* to a FO formula ψ with n free variables, in symbols $\psi \equiv_s u$, whenever u' and ψ are FO equivalent. Two λ-FO expressions $u = \lambda\alpha_1 \ldots \lambda\alpha_k.\lambda P_1 \ldots \lambda P_m.\lambda v_{i_1} \ldots \lambda v_{i_n}.u''$ and $u' = \lambda\alpha'_1 \ldots \lambda\alpha'_{k'}.\lambda P'_1 \ldots \lambda P'_{m'}.\lambda v'_{i_1} \ldots \lambda v'_{i_{n'}}.u'''$ are said to *equivalent*, in symbols $u \equiv u'$, whenever u'' and u''' are FO equivalent. Clearly, if u is a FO formula (or sentence), equivalence and structural equivalence coincide. The FO standard substitution and replacement properties trivially hold for equivalence thus understood.

THEOREM 5.1 (Lite-English). (a) *For each sentence D in Lite-English, there exists an assertion σ s.t. $\tau(D) \equiv \sigma^{t_x}$.* (b) *For each DL-Lite$_\sqcap$ assertion σ, there exists a sentence D in Lite-English s.t. $\tau(D) \equiv \sigma^{t_x}$.*

PROOF SKETCH. To prove Claim (a), we need to show that, for $f \in \{l, r\}$:

$$\begin{array}{c}\text{for each Lite-English } \mathbf{VP}_f \text{ or } \mathbf{N}_f \text{ constituent, there exists}\\ \text{a } \textit{DL-Lite}_\sqcap \text{ concept } C_f \text{ s.t. } \tau(\mathbf{VP}_f) \equiv_s {C_f}^{t_x} \text{ or } \tau(\mathbf{N}_f) \equiv_s {C_f}^{t_x},\end{array} \qquad (\dagger)$$

by mutual induction on the length of derivations rooted in $\mathbf{VP}_f$s and $\mathbf{N}_f$s.

(Basis) There are eight possibilities. Either $\mathbf{N}_f \Longrightarrow \mathbf{N}$, $\mathbf{VP}_f \Longrightarrow \mathbf{IV}$, $\mathbf{VP}_f \Longrightarrow$ is a $\mathbf{N}$, or $\mathbf{VP}_f \Longrightarrow$ is $\mathbf{Adj}$, for $f \in \{l, r\}$. In all eight cases $\tau(\cdot)$ maps them to $\lambda x.A(x)$, where $A(x)$ is a concept name or atomic formula.

(Inductive step) We look at one case only, the argument is similar for all the remaining ones. $\mathbf{N}_f \Longrightarrow^{k+1} \mathbf{N}_f\ \mathbf{Relp}_f \Longrightarrow^{k} \mathbf{N}\ \mathbf{Relp}_f\ \mathbf{VP}_f$, for $f \in \{l, r\}$. Now $\mathbf{VP}_f \Longrightarrow^{k-1} w''$ for some sequence w'' of terminal and non-terminal symbols. By IH, on derivations of length $\leq k$ rooted in $\mathbf{VP}_f$, $\tau(\mathbf{VP}_f) \equiv_s {C''_f}^{t_x}$, for some formula ${C''_f}^{t_x}$. On the other hand, $\mathbf{N}_f \Longrightarrow^{k-1} w'$; whence $\tau(\mathbf{N}_f) \equiv_s {C'_f}^{t_x}$, again by IH.
Therefore, $\tau(\mathbf{N}_f) := (\tau(\mathbf{Relp}_f)(\tau(\mathbf{N}_f)))(\tau(\mathbf{VP}_f)) \equiv_{ih} \lambda P.\lambda Q.\lambda x.(Q(x) \wedge P(x))(\lambda x.{C'_f}^{t_x})(\lambda x.{C''_f}^{t_x}) \rhd \lambda x.{C'_f}^{t_x} \wedge {C''_f}^{t_x}$, and $\tau(\mathbf{N}_f) \equiv_s {C'_f}^{t_x} \wedge {C''_f}^{t_x}$, the FO translation of a left or right concept.

We are now ready to associate a complete meaning representation to each Lite-English sentence D. We have two cases to consider: $\mathbf{S} \Longrightarrow^* \mathbf{Det}_l\ \mathbf{N}_l\ \mathbf{VP}_r$ and $\mathbf{S} \Longrightarrow^* \mathbf{Pro}_l\ \mathbf{Relp}_l\ \mathbf{VP}_r$. In both cases, modulo (†) is is easy to see that $\tau(\mathbf{S}) \equiv \forall x({C_l}^{t_x} \Rightarrow {C_r}^{t_x})$.

To prove Claim (b), we need to show that, for $f \in \{l, r\}$:

$$\begin{array}{c}\text{for each } \textit{DL-Lite}_\sqcap \text{ concept } C_f\text{, there exists a Lite-English constituent } w\\ \text{s.t. } \mathbf{VP} \Longrightarrow^* w \text{ and } \tau(w) \equiv_s {C_f}^{t_x}\text{, or } \mathbf{N} \Longrightarrow^* w \text{ and } \tau(w) \equiv_s {C_f}^{t_x},\end{array} \qquad (\ddagger)$$

by (a tedious, albeit simple) structural induction on formulas ${C_f}^{t_x}$. We do it only for left formulas. Moreover, we will deal with only one of the inductive cases. The proof for right concepts and for the other cases proceeds analogously.

(Basis) There are two possibilities. Either ${C_l}^{t_x} = A(x)$, for which we consider $\mathbf{N}_l \Longrightarrow^* \mathbf{N} \Longrightarrow^* A$ (resp., $\mathbf{VP}_l \Longrightarrow^*$ is a $\mathbf{N} \Longrightarrow^*$ is a A), or ${C_l}^{t_x} = \exists y R(x, y)$,for which we consider $\mathbf{VP}_l \Longrightarrow^* \mathbf{TV}\ \mathbf{NP}_l \Longrightarrow^* \mathbf{TV}\ \mathbf{Pro}_l \Longrightarrow^*$ Rs somebody. Clearly, $\tau(A) \equiv_s A$ and $\tau(R\text{s somebody}) \equiv_s \exists y R(x, y)$.

(Inductive step) ${C_l}^{t_x} = {C'_l}^{t_x} \wedge {C''_l}^{t_x}$. There are four sub-cases. We consider only one. By IH there exists a $\mathbf{VP}'_l$ and a $\mathbf{VP}''_l$ s.t. $\mathbf{VP}'_l \Longrightarrow^* w'$ and $\mathbf{VP}''_l \Longrightarrow^* w''$ with, $\tau(\mathbf{VP}'_l) \equiv_s {C'_l}^{t_x}$ and $\tau(\mathbf{VP}''_l) \equiv_s {C''_l}^{t_x}$, respectively. The desired constituent w rooted in $\mathbf{VP}_l$ is obtained via $\mathbf{VP}_l \Longrightarrow^* \mathbf{VP}'_l\ \mathbf{Crd}\ \mathbf{VP}''_l \Longrightarrow^* w'$ and w''. Clearly, $\tau(w) = \tau(w' \text{ and } w'') \equiv_s {C'_l}^{t_x} \wedge {C''_l}^{t_x}$. Similarly for the other cases.

We now need to show that, when such concepts or formulas are put together into assertions, such assertions can be captured by a Lite-English sentence. Let $\sigma = \forall x(C_l^{t_x} \Rightarrow C_r^{t_x})$ be a *DL-Lite*$_\sqcap$ assertion. There are two possibilities: either *(i)* $\mathbf{S} \Longrightarrow^* \mathbf{Det}_l\ \mathbf{N}_l\ \mathbf{VP}_r \Longrightarrow^*$ no/every $w\ w'$, or *(ii)* $\mathbf{S} \Longrightarrow^* \mathbf{Pro}_l\,\mathbf{Relp}_l\,\mathbf{VP}_f\,\mathbf{VP}_r \Longrightarrow^*$ anybody who $\mathbf{VP}_l\,\mathbf{VP}_r \Longrightarrow^*$ anybody who $w\,w'$. Modulo (‡), it follows that *(i)* "no/every $\mathbf{N}_l\ \mathbf{VP}_r$", or *(ii)* "anybody who $\mathbf{VP}_f\ \mathbf{VP}_r$" are the desired Lite-English sentences (or sentence patterns). ■

EXAMPLE 5.2. By considering the content lexicon shown in Figure 2, we can generate sentences like:

Every man loves somebody. (1)

Anybody who loves somebody likes somebody. (2)

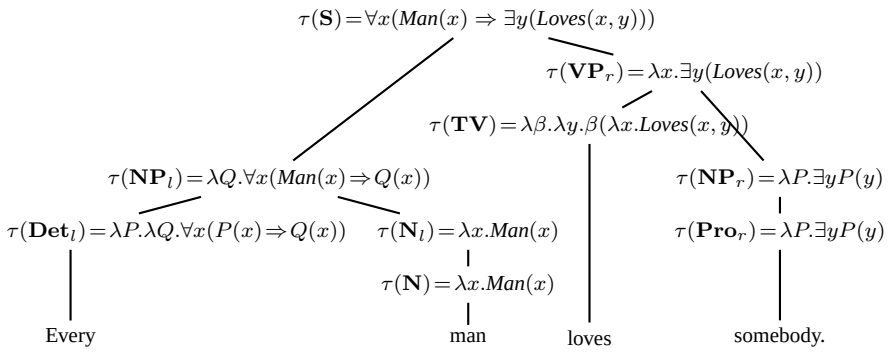
Figure 2 shows that sentence (1) is indeed a Lite-English declarative sentence expressing the FO formula $\forall x(\textit{Man}(x) \Rightarrow \exists y(\textit{Loves}(x, y)))$, which corresponds to the *DL-Lite*$_\sqcap$ ontology assertion $\textit{Man} \sqsubseteq \exists \textit{Loves}$. At each node, the meaning representation built is the (beta) reduct of its immediate successors, down to the yield.

On the other hand, English sentences like:

*Somebody who loves anybody is a happy man. (3)

do not belong to Lite-English. Why? Because *(i)* "somebody" cannot occur in subject position, and *(ii)* "anybody" cannot occur in predicate position. Overgeneration is thus prevented by types and λ-FO typing rules. For instance, the *right* $\mathbf{VP}$ "loves anybody" will be discarded because the pronoun "anybody" is typed in Lite-English as a determiner of type $(e{\rightarrow}t){\rightarrow}((e{\rightarrow}t){\rightarrow}t)$, but transitive verbs take generalized quantifiers of type $(e{\rightarrow}t){\rightarrow}t$ as arguments: "loves" cannot combine with "anybody" because $\tau(\text{loves})$ *cannot* be applied to $\tau(\text{anybody})$. ♣

GCQ-English. GCQs are captured by the interrogative fragment GCQ-English. Questions in GCQ-English fall in two main classes : *(i)* Wh-questions, which will map into non-Boolean GCQs, and *(ii)* Y/N-questions, which will map into Boolean



Content Lexicon	
$\mathbf{N} ::=$ woman	$\tau(\mathbf{N}) := \lambda x.\mathit{Woman}(x)$
$\mathbf{N} ::=$ man	$\tau(\mathbf{N}) := \lambda x.\mathit{Man}(x)$
$\mathbf{Adj} ::=$ happy	$\tau(\mathbf{Adj}) := \lambda x.\mathit{Happy}(x)$
$\mathbf{TV} ::=$ loves	$\tau(\mathbf{TV}) := \lambda\beta.\lambda x.\beta(\lambda y.\mathit{Loves}(x,y))$
$\mathbf{TV} ::=$ likes	$\tau(\mathbf{TV}) := \lambda\beta.\lambda x.\beta(\lambda y.\mathit{Likes}(x,y))$

Figure 2. **Above:** Translating "Every man loves somebody." expresses the *DL-Lite*$_\sqcap$ assertion $\mathit{Man} \sqsubseteq \exists \mathit{Loves}$. **Below:** A sample content lexicon for Lite-English, denoting individuals, sets and relations. The number of content words can be arbitrarily large but there can be only finitely many function words.

GCQs. Table 3 shows GCQ-English's grammar G_{GCQ}. Some basic morphosyntactic and semantic features are attached to (some) constituents. The feature $\cdot^-$ means that the constituent is of negative polarity. Absence of features indicates that constituents are of positive polarity. Notice that, as for Lite-English, we disregard all other morphosyntactic features.

GCQ-English covers personal, reflexive, and relative pronouns and can capture a restricted form of *anaphora*, the phenomenon by which pronouns co-refer with the $\mathbf{NP}$s that dominate them and that occur earlier in the utterance. Personal pronouns (e.g., "him") co-refer with the closest $\mathbf{NP}$ in *argument* position. Reflexive pronouns (e.g., "himself"), like relative pronouns, co-refer with their closest $\mathbf{NP}$ in *subject* position. The co-reference of pronouns is captured by indexing constituents. A reflexive pronoun (resp., a personal pronoun) dominated by a $\mathbf{NP}$ of index i co-refers with an $\mathbf{NP}$ of index $i-1$ (resp., an $\mathbf{NP}$ of index $i-2$).

EXAMPLE 5.3. Consider again the content lexicon in Figure 2. The English question:

Who loves some woman who likes somebody? (4)

belongs to GCQ-English and expresses the GCQ $\exists y(\mathit{Loves}(x,y) \wedge \mathit{Woman}(y) \wedge$

Table 3. Phrase structure rules and function lexicon of GCQ-English. Note the use of indexes to resolve co-references. For simplicity, we use empty noun phrases ϵ as subjects of subordinated sentences. Notice the non-standard meaning representation of "someone".

Rules and Semantic Actions G_{GCQ}	
$\mathbf{Q}_{wh} ::= \mathbf{Intpro}\ \mathbf{N}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \tau(\mathbf{Intpro})(\tau(\mathbf{N}_i))(\tau(\mathbf{S}_{g_i}))$
$\mathbf{Q}_{wh} ::= \mathbf{Intpro}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \tau(\mathbf{Intpro}_i)(\tau(\mathbf{S}_{g_i}?))$
$\mathbf{Q}_{Y/N} ::=$ does $\mathbf{NP}_i^-\ \mathbf{VP}_i^-?$	$\tau(\mathbf{Q}_{Y/N}) := \tau(\mathbf{NP}_i^-)(\tau(\mathbf{VP}_i^-))$
$\mathbf{Q}_{Y/N} ::=$ is $\mathbf{NP}_i\ \mathbf{VP}_i$	$\tau(\mathbf{Q}_{Y/N}) := \tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i))$
$\mathbf{S}_{g_i} ::= \mathbf{NP}_{g_i}\mathbf{VP}_i$	$\tau(\mathbf{S}_{g_i}) := \tau(\mathbf{NP}_{g_i})(\tau(\mathbf{VP}_i))$
$\mathbf{N}_i ::= \mathbf{Adj}\ \mathbf{N}_i$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_i))$
$\mathbf{N}_i ::= \mathbf{N}_i\ \mathbf{Relp}_i\ \mathbf{S}_{g_i}$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Relp}_i)(\tau(\mathbf{N}_i))(\tau(\mathbf{S}_{g_i}))$
$\mathbf{VP}_i ::=$ is $\mathbf{Adj}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{Adj}_i)$
$\mathbf{VP}_i ::=$ is a $\mathbf{N}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{N}_i)$
$\mathbf{VP}_i ::= \mathbf{VP}_i\ \mathbf{Crd}\ \mathbf{VP}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}_i))(\tau(\mathbf{VP}_i))$
$\mathbf{VP}_i^- ::= \mathbf{VP}_i^-\ \mathbf{Crd}\ \mathbf{VP}_i^-$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}_i^-))(\tau(\mathbf{VP}_i^-))$
$\mathbf{VP}_i ::= \mathbf{IV}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{IV}_i)$
$\mathbf{VP}_i^- ::= \mathbf{IV}_i^-$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{IV}_i^-)$
$\mathbf{VP}_i ::= \mathbf{TV}_{i,i+1}\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{TV}_{i,i+1})(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{VP}_i^- ::= \mathbf{TV}_{i,i+1}^-\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{TV}_{i,i+1}^-)(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{NP}_i ::= \mathbf{Pro}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Pro}_i)$
$\mathbf{NP}_i^- ::= \mathbf{Pro}_i^-$	$\tau(\mathbf{NP}_i^-) := \tau(\mathbf{Pro}_i^-)$
$\mathbf{NP}_i ::= \mathbf{Det}\ \mathbf{N}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Det})(\tau(\mathbf{N}_i))$
$\mathbf{NP}_i ::= \mathbf{Pn}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Pn}_i)$
$\mathbf{NP}_{g_i} ::= \epsilon$	$\tau(\mathbf{NP}_{g_i}) := \lambda P.P$
$\mathbf{Det} ::=$ some	$\tau(\mathbf{Det}) := \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
$\mathbf{Pro}_i ::=$ somebody	$\tau(\mathbf{Pro}_i) := \lambda P.\exists x P(x)$
$\mathbf{Pro}_i^- ::=$ anybody	$\tau(\mathbf{Pro}_i^-) := \lambda P.\exists x.P(x)$
$\mathbf{Crd} ::=$ and	$\tau(\mathbf{Crd}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$\mathbf{Relp}_i ::=$ who	$\tau(\mathbf{Relp}_i) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$\mathbf{Intpro} ::=$ which	$\tau(\mathbf{Intpro}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$\mathbf{Intpro}_i ::=$ who	$\tau(\mathbf{Intpro}_i) := \lambda P.\lambda x.P(x)$
$\mathbf{Pro}_{i-2} ::=$ him	$\tau(\mathbf{Pro}_{i-2}) := \lambda P.P(x)$
$\mathbf{Pro}_{i-1} ::=$ himself	$\tau(\mathbf{Pro}_{i-1}) := \lambda P.P(x)$

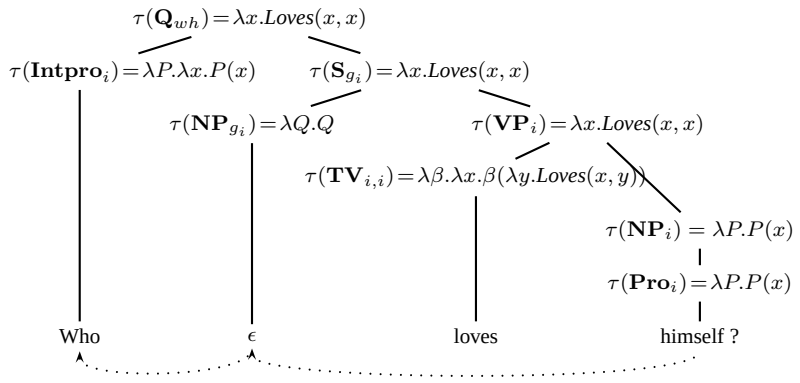


Figure 3. Translating "Who loves himself?" into the GCQ $\mathit{Loves}(x,x)$. Note how indexes capture co-references, since $i = (i+1) - 1$.

$\exists z(\mathit{Likes}(y,z)))$. Likewise, the questions:

Does somebody love some man who loves him? (5)

Who loves himself? (6)

are GCQ-English questions. The personal pronoun "him" co-refers with "some man"; thus, Y/N-question (5) translates into the GCQ $\exists x(\exists y(\mathit{Loves}(x,y) \wedge \mathit{Man}(y) \wedge \mathit{Loves}(y,x)))$. Similarly, the pronoun "himself" co-refers with "who", and question (6) translates (up to structural equivalence) into the GCQ $\mathit{Loves}(x,x)$ (see Figure 3). On the other hand:

*Which man loves some woman who likes some woman who loves him? (7)

lies outside this controlled language (and does not express a GCQ). Why? Because "him" cannot co-refer with "which man": co-reference cannot span beyond the most immediate nesting phrase. ♣

THEOREM 5.4 (Expressing GCQs). (a) *For every GCQ-English question Q, there exists a GCQ φ s.t. $\tau(Q) \equiv_s \varphi$.* (b) *For every GCQ φ, there exists a GCQ-English question Q s.t. $\tau(Q) \equiv_s \varphi$.*

PROOF SKETCH. The proof of Claim (a) is similar to the proof of Claim (a) of Theorem 5.1. We show, by mutual induction on the length of derivations rooted in $\mathbf{N}_i$s and $\mathbf{VP}_i$s, that:

for each $\mathbf{N}_i$ or $\mathbf{VP}_i$ GCQ-English constituent,
there exists a GCQ $\varphi(x)$ s.t. $\tau(\mathbf{N}_i) \equiv_s \varphi(x)$ or $\tau(\mathbf{VP}_i) \equiv_s \varphi(x)$, resp. (†)

(Basis) Trivial.

(Inductive step) We show only one case, as all the other cases are analogous. Let $\mathbf{VP}_i \Longrightarrow^{k+1} \mathbf{TV}_{i,i+1}\,\mathbf{NP}_{i+1} \Longrightarrow^{k} \mathbf{TV}_{i,i+1}\,\mathbf{Det}\,\mathbf{N}_{i+1} \Longrightarrow^{k-1} R\text{s some } w$, with $\mathbf{N}_i \Longrightarrow^{k-2} w$. By IH, $\tau(\mathbf{N}_i) \equiv_s \varphi(x)$. Thus, we have that $\tau(\mathbf{VP}_i) := \lambda\beta.\lambda x.\beta(\lambda y.R(x,y))(\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z)))(\tau(\mathbf{N}_i)) \equiv_{ih} \lambda\beta.\lambda x.\beta(\lambda y.R(x,y))(\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z)))(\lambda z\varphi(z)) \rhd \lambda x.\exists y(R(x,y) \wedge \varphi(y))$, which is structurally equivalent to a GCQ.

With Claim (†) established, we can consider full questions. Since the argument is similar both for Y/N and Wh-questions, we only deal with one of the four possible cases. Let $\mathbf{Q}_{wh} \Longrightarrow \mathbf{Intpro}\,\mathbf{N}_i\,\mathbf{S}_{g_i} \Longrightarrow \mathbf{Intpro}\,\mathbf{N}_i\,\mathbf{VP}_i \Longrightarrow^* \text{which } w'\,w''$. Therefore, $\tau(\mathbf{Q}_{wh}) := \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))(\lambda y.\varphi'(y))(\lambda z.\varphi''(z)) \rhd \lambda x.\varphi'(x) \wedge \varphi''(x)$, which is structurally equivalent to a GCQ.

The proof of Claim (b) is similar to the proof of Claim (b) of Theorem 5.1. We prove, by induction on $\varphi(x)$, a non-Boolean GCQ φ of distinguished variable x, that:

$$\begin{array}{c}\text{for each GCQ } \varphi(x), \text{ there exists a GCQ-English constituent } w \text{ s.t.} \\ \mathbf{VP}_i \Longrightarrow^* w \text{ and } \tau(w) \equiv_s \varphi(x), \text{ or } \mathbf{N}_i \Longrightarrow^* w \text{ and } \tau(w) \equiv_s \varphi(x).\end{array} \quad (\ddagger)$$

(Basis) There are four possibilities. If $\varphi(x) = A(x)$, then we consider $\mathbf{N}_i \Longrightarrow^* A$ (resp., $\mathbf{VP}_i \Longrightarrow^*$ is a $\mathbf{N}_i \Longrightarrow^*$ is a A). If $\varphi(x) = R(x,x)$, we consider $\mathbf{VP}_i \Longrightarrow^* \mathbf{TV}_{i,i}\,\mathbf{NP}_i \Longrightarrow^* \mathbf{TV}_{i,i}\,\mathbf{Pro}_i \Longrightarrow^* R\text{s himself}$. If $\varphi(x) = R(x,c)$, we consider $\mathbf{VP}_i \Longrightarrow^* \mathbf{TV}_{i,i+1}\,\mathbf{NP}_{i+1} \Longrightarrow^* \mathbf{TV}_{i,i+1}\,\mathbf{Pn}_{i+1} \Longrightarrow^* R\text{s } c$. If $\varphi(x) = \exists y(R(x,y))$, we consider $\mathbf{VP}_i \Longrightarrow^* \mathbf{TV}_{i,i+1}\,\mathbf{NP}_{i+1} \Longrightarrow^* \mathbf{TV}_{i,i+1}\,\mathbf{Pro}_{i+1} \Longrightarrow^* R\text{s somebody}$. Now, $\tau(A) \equiv_s A(x)$, $\tau(R\text{s himself}) \equiv_s R(x,x)$, $\tau(R\text{s } c) \equiv_s R(x,c)$ and, finally, $\tau(R\text{s somebody}) \equiv_s \exists y(R(x,y))$.

(Inductive step) Again, we look at one case only. Let $\varphi(x) = \exists y(S(x,y) \wedge \varphi'(y))$. By IH there exists a GCQ-English constituent w' rooted in a $\mathbf{VP}'_{i+1}$ or a nominal $\mathbf{N}'_{i+1}$ s.t. $\tau(w') \equiv_s \varphi'(y)$. This gives rise to several alternative GCQ-English constituents rooted in category $\mathbf{VP}_i$. We consider only one possible case. Notice that we need to show by an easy induction on relations, that there exists a GCQ-English constituent w'' s.t. $\tau(w'') \equiv_s S(x,y)$. We consider only the base case, where $S(x,y) = R(x,y)$. Then, $\mathbf{VP}_i \Longrightarrow^* \mathbf{TV}_{i,i+1}\mathbf{NP}_{i+1} \Longrightarrow^* \mathbf{TV}_{i,i+1}\mathbf{Det}\,\mathbf{N}_{i+1} \Longrightarrow^* R\text{s some } w'$. Whence, $\tau(R\text{s some } w') \equiv_s \exists y(R(x,y) \wedge \varphi'(y))$.

Full Boolean and non-Boolean GCQs are captured in several, alternative ways by GCQ-English questions. As the argument is similar for all cases, we deal only with Boolean GCQs and exhibit one (of possibly many) GCQ-English derivations. Consider a Boolean GCQ $\varphi = \exists x(\varphi'(x) \wedge \varphi''(x))$. If $\mathbf{Q}_{Y/N} \Longrightarrow^*$ is $\mathbf{NP}_i\,\mathbf{VP}_i \Longrightarrow^*$

is $\mathbf{Det}\,\mathbf{N}_i\,\mathbf{VP}_i \Longrightarrow^*$ is some $\mathbf{N}_i\,\mathbf{VP}_i$?, then, by Claim (‡), $\mathbf{N}_i \Longrightarrow^* w'$ with $\tau(w) \equiv_s \varphi'(x)$, and $\mathbf{VP}_i \Longrightarrow^* w''$ with $\tau(w) \equiv_s \varphi''(x)$. This means that "is some $w'\,w''$?" is the desired (controlled) question: Firstly, $\mathbf{Q}_{Y/N} \Longrightarrow^*$ is some $w'\,w''$?. Secondly, "is" and "?" have no meaning by themselves. Thus $\tau(\text{is some } w'\,w''\,?) = \tau(\text{some } w'\,w'') = \tau(\text{some})(\tau(w'))(\tau(w'')) \equiv \exists x(\varphi'(x) \wedge \varphi''(x)) = \varphi$. This concludes the proof. ■

Data Complexity Results. Lite-English in combination with GCQ-English gives rise to "optimal" data complexity for KBQA. What we mean by this is that, in this case, KBQA reduces to plain database query evaluation (thus allowing to exploit in theory all the optimizations available for relational databases).

THEOREM 5.5. *KBQA is in* AC^0 *w.r.t. data complexity for Lite-English and GCQ-English.*

PROOF. It is shown in [2], Theorem 7.1, that KBQA is in AC^0 in data complexity for *DL-Lite*$^{\mathcal{R}}_{horn}$ knowledge bases and CQs. The *DL-Lite*$^{\mathcal{R}}_{horn}$ logic strictly contains *DL-Lite*$_{\sqcap}$ and CQs strictly contain GCQs by definition. Since we have shown that Lite-English expresses *DL-Lite*$_{\sqcap}$ (Theorem 5.1) and GCQ-English GCQs (Theorem 5.4), the result follows. ■

6. The family $\{\text{IS-A}_i\}_{i\in[0,7]}$ of Controlled Languages

In this section we study a family of controlled fragments, the $\{\text{IS-A}_i\}_{i\in[0,7]}$ family, which show the boundary between maximally tractable and minimally intractable controlled languages for data access, when considered together with queries. The results in this section generalize results from [19]. We consider CQs when deriving data complexity upper bounds and GCQs (and fragments) when deriving lower bounds. These fragments are also obtained by conveniently combining the grammar rules from Table 1 and by consequently constraining what can be said in the subject $\mathbf{NP}$ and in the predicate $\mathbf{VP}$ of complete sentences. Specifically, each fragment IS-A_i is generated by the grammar whose phrase structure rules are the union of $G_{\sqsubseteq} \cup G_{A_l} \cup G_{A_r}$ with the rules shown in Table 4. Each set of phrase structure rules and function lexicons can be clearly combined with an appropriate content lexicon.

Each IS-A_i fragment generates a fragment of $\mathcal{ALCI}$ that might contain, besides inclusion assertions that are allowed in *DL-Lite*$_{\sqcap}$, also inclusion assertions σ_i of the form shown in Table 5, which go beyond what can be expressed in *DL-Lite*$_{\sqcap}$, and are responsible for the complexity lower bounds given in Table 4, as shown next.

Table 4. Grammars and computational complexity of KBQA for the $\{\text{IS-A}_i\}_{i\in[0,7]}$ family. Each fragment IS-A_i is generated by the rules $G_{\sqsubseteq} \cup G_{A_l} \cup G_{A_r}$ combined with the rules shown in the grammar column.

	Grammar	**Data Compl.**
IS-A_1	$G_{(\forall R{:}C_r)_r}$	NLSPACE-complete
IS-A_2	$G_{(\forall R{:}C_r)_r} \cup G_{(C_l \sqcap C'_l)_l} \cup G_{(C_r \sqcap C'_r)_r}$	PTIME-complete
IS-A_3	$G_{(\exists R{:}C_l)_l} \cup G_{(\exists R^-{:}C_l)_l} \cup G_{(C_r \sqcap C'_r)_r} \cup G_{(\exists R)_r}$	PTIME-complete
IS-A_4	$G_{(\exists R{:}C_l)_l} \cup G_{(C_l \sqcap C'_l)_l} \cup G_{(C_r \sqcap C'_r)_r}$	PTIME-complete
IS-A_5	$G_{(\forall R{:}C_l)_l} \cup G_{(C_r \sqcap C'_r)_r}$	CONP-complete
IS-A_6	$G_{(C_r \sqcup C'_r)_r}$	CONP-complete
IS-A_7	$G_{(\neg C_r)_r}$	CONP-complete

Data Complexity Results. We state now the main data complexity results for the $\{\text{IS-A}_i\}_{i\in[1,7]}$ fragments. All of them are orthogonal in expressive power to each other and, most importantly, to Lite-English and *DL-Lite*$_\sqcap$, covering English constructs and expressing concepts none of the latter can. For all of them, KBQA lies *beyond* AC^0. In particular, note that CONP-hardness arises for those fragments that are powerful enough to express Boolean satisfiability (which we may accordingly term "Boolean-closed"). Table 5 spells out for each IS-A_i fragment the assertions that cause the increase in complexity, and gives some examples of utterances, while Table 4 summarizes the complexity results for the IS-A_i fragments.

THEOREM 6.1. *KBQA is* (a) PTIME*-complete w.r.t. data complexity for IS-*A_2*, IS-*A_3*, IS-*A_4 *and (G)CQs, and* (b) CONP*-complete w.r.t. data complexity for IS-*A_5*, IS-*A_6*, IS-*A_7 *and (G)CQs.*

PROOF. The lower bounds for IS-A_2, IS-A_3 and IS-A_4 follow from the results in [6]. For IS-A_2 the result is derived from Theorem 7, case 2. For IS-A_3, it is derived from Theorem 6, case 1. Finally, the lower bound for IS-A_4 follows from Theorem 7, case 3. Basically, this is because our controlled languages subsume the description logics for which those theorems hold. PTIME-hardness in all three cases holds already for *atomic queries*, viz., GCQs of the form $A(c)$ or $R(c, c')$. The complexity upper bounds, on the other hand, follow from results in [12] for answering CQs in the description logic $\mathcal{EL}$, which subsumes the description logic assertions that IS-A_2, IS-A_3, and IS-A_4 express. For IS-A_4 we need to observe that each assertion $A \sqsubseteq \forall R{:}A'$ can be replaced by the equivalent assertion $\exists R^-{:}A \sqsubseteq A'$. Moreover, after such a replacement, only inverse roles are present in the resulting TBox, so each inverse role R^- can be replaced by a new

Table 5. The $\{\text{IS-A}_i\}_{i\in[1,7]}$ controlled languages.

	C-hard **Assertion(s)** σ_i	**Sentence(s)** D_i
IS-A_1	$A \sqsubseteq \forall R{:}A'$	Every herbivore eats only herbs.
IS-A_2	$A \sqcap A' \sqsubseteq \forall R{:}(A'' \sqcap A''')$	Every Italian man drinks only strong coffee.
IS-A_3	$\exists R{:}A \sqsubseteq A \sqcap A'$	Anybody who murders some person is a heartless killer.
	$\exists R^{-}{:}A \sqsubseteq A \sqcap A'$	Anybody who is loved by some person is a happy person.
	$A \sqsubseteq \exists R$	Every driver drives something.
IS-A_4	$A \sqcap A' \sqsubseteq A'' \sqcap A'''$	Every cruel man is a bad man.
	$\exists R{:}(A \sqcap A') \sqsubseteq A'' \sqcap A'''$	Anybody who runs some bankrupt company is a bad businessman.
IS-A_5	$\forall R{:}A \sqsubseteq A' \sqcap A''$	Anybody who values only money is a greedy person.
IS-A_6	$A \sqsubseteq A' \sqcup A''$	Every mammal is male or is female.
IS-A_7	$\neg A \sqsubseteq A'$	Anybody who is not selfish is altruistic.

role R_{inv}, which results in an $\mathcal{EL}$ TBox. The lower bounds for IS-A_5, IS-A_6, and IS-A_7 follow also from [6]: for IS-A_5, we apply Theorem 8, case 3; for IS-A_6, we apply Theorem 8, case 2; and for IS-A_7, Theorem 8, case 1. The CONP upper bounds for these fragments, on the other hand, derive from the CONP data complexity upper bounds for KBQA over expressive description logics shown in [14] and hold, again, for CQs. ■

THEOREM 6.2. *KBQA is* NLSPACE*-complete w.r.t. data complexity for IS-*A_1 *and (G)CQs.*

PROOF. For the hardness part, it is shown in [6] (by a reduction of the reachability problem for directed graphs) that for any description logic capable of expressing assertions of the form $A \sqsubseteq \forall R{:}A'$, or, equivalently, of the form $\exists R^{-}{:}A \sqsubseteq A'$, KBQA is NLSPACE-hard w.r.t. data complexity. This result holds already for atomic queries. Note that such assertions are expressed in our fragments by sentences of the form "Every A Rs only A's".

For the membership part, let $\varphi := \exists \bar{y}(\varphi'(\bar{x}, \bar{y}))$ be a *fixed* CQ, $\bar{c}$ a *fixed* tuple, $\mathcal{O}$ a *fixed* set of universally quantified IS-A_1 meaning representations, and $\mathcal{D}$ a set of facts. We sketch a data complexity reduction of KBQA for IS-A_1 to KBQA for linear Datalog, which is known to be NLSPACE-complete in data complexity (see [7], Theorem 4.3). Such a reduction will be space logarithmic in $|\mathit{adom}(\mathcal{D})|$

(and time polynomial in $|\mathit{adom}(\mathcal{D})|$). By combining the reduction with a query answering algorithm for linear Datalog, we derive the desired result. The correctness of the reduction entails the correctness of the overall non-deterministic (space logarithmic in $|\mathit{adom}(\mathcal{D})|$) decision procedure.

A Datalog clause is a disjunction $\pm S_1(\bar{z}_1) \vee \neg S_2(\bar{z}_2) \vee \cdots \vee \neg S_n(\bar{z})$. A relational atom or clause with no free variables is said to be *ground*. If a clause contains no positive (i.e., non-negated) relational atom, it is called a *goal*; if it contains no negative (i.e., negated) relational atom(s), it is called a *fact*; and if it contains both a positive and at least one negative relational atoms, it is called a *rule*. Given a clause $S_1(\bar{z}_1) \vee \neg S_2(\bar{z}_2) \vee \cdots \vee \neg S_n(\bar{z}_n)$, $S_1(\bar{z}_1)$ is said to be the clause's *head* and $\neg S_2(\bar{z}_2) \vee \cdots \vee \neg S_n(\bar{z}_n)$ the clause's *body*. Every atom occuring as head of a Datalog clause is said to be *intensional*. A set of Datalog clauses is called a *program*. A *linear* Datalog clause is a Datalog clause in which intensional atoms are allowed to occur *at most once* in the clause's body.

The only inclusion assertions expressible by the IS-A_1 fragment are of the form $A \sqsubseteq A'$ and $\exists R{:}A \sqsubseteq A'$, which can be transformed into a set $\mathcal{P}_\mathcal{O}$ of linear Datalog clauses of the form $\neg A(x) \vee A'(x)$ and $\neg R(x,y) \vee \neg A(y) \vee A'(x)$, respectively. Since $\mathcal{O}$ is fixed, transforming it into $\mathcal{P}_\mathcal{O}$ does not affect data complexity. Moreover, $\mathcal{D}$ is already a set of linear Datalog clauses (a set of gound facts). Now, the CQ φ is not a linear Datalog goal. However, $\varphi'(\bar{x},\bar{y})$ consists of a conjunction of k relational atoms $S_1(\bar{z}_1) \wedge \cdots \wedge S_k(\bar{z}_k)$, where $\bar{x} \cup \bar{y} = \bar{z}_1 \cup \cdots \cup \bar{z}_k$. Ground φ by $\theta_{\bar{c}} := \{\bar{x} \mapsto \bar{c}\}$, which returns (the Boolean CQ) $\exists \bar{y}(\varphi'(\bar{c},\bar{y}))$. This does not affect, once again, data complexity. Next, consider all the possible groundings $\theta_{\bar{c}'} := \{\bar{y} \mapsto \bar{c}'\}$ with $\bar{c}' \in \mathit{adom}(\mathcal{D})^{|\bar{y}|}$. There are $\mathbf{O}(|\mathcal{D}|^{|\bar{y}|})$ such groundings, and each of them can be stored in a register of $\mathbf{O}(\log|\mathcal{D}|)$ size. Applying a $\theta_{\bar{c}'}(\cdot)$ grounding to $\varphi'(\bar{c},\bar{y})$ yields a family of CQs $\varphi'(\bar{c},\bar{c}') = S_1(\bar{c}''_1) \wedge \cdots \wedge S_k(\bar{c}''_k)$, which can be stored (because the query is fixed) in a registry of $\mathbf{O}(\log|\mathcal{D}|)$ size. We now claim that:

$$(\mathcal{O},\mathcal{D}) \models \exists \bar{y}(\varphi'(\bar{c},\bar{y})) \quad \text{iff} \quad \begin{array}{l}\text{there exists a } \theta_{\bar{c}'} \text{ s.t., for all } i \in [1,k], \\ \mathcal{P}_\mathcal{O} \cup \mathcal{D} \models S_i(\bar{c}''_i) \text{ and } \bar{c}''_i \subseteq \bar{c}' \cup \bar{c}''.\end{array} \qquad (\dagger)$$

The "if" direction is immediate. To prove the "only-if" direction, we reason as follows. Assume for contradiction that there exists an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{P}_\mathcal{O} \cup \mathcal{D}$, but for every assignment $\theta_{\bar{c}'}(\cdot)$ we have that $\mathcal{I} \not\models S_i(\bar{c}''_i)$, for some $i \in [1,k]$, with $\bar{c}''_i \subseteq \bar{c} \cup \bar{c}'$. Since $\mathcal{I} \models \mathcal{P}_\mathcal{O} \cup \mathcal{D}$, we have that $\mathcal{I} \models (\mathcal{O},\mathcal{D})$ and $\mathcal{I} \models \exists \bar{y}(\varphi'(\bar{c},\bar{y}))$. By FO semantics, this implies that there exists some grounding $\theta(\cdot)$ from $\bar{y}$ into $\mathit{adom}(\mathcal{D})$, s.t. $\mathcal{I} \models S_i(\bar{z}_i)\theta$. But then, as $\theta(\cdot)$ is among the $\theta_{\bar{c}'}(\cdot)$s, this implies that $\mathcal{I} \models S_i(\bar{c}''_i)$ as well. Contradiction.

The algorithm then proceeds by looping k times over the $S_i(\bar{c}''_i)$s (stored in the $\mathbf{O}(\log|\mathcal{D}|)$ registry), each time running a linear Datalog non-deterministic check

Table 6. Controlled English constructs for which (G)CQ query answering is respectively tractable and intractable in data complexity.

Complexity	**Constructs**	**Fragment(s)**
Tractable	Negation (“not”) in predicate position. Relatives (“who”, “which”) everywhere. Conjunction (“and”) everywhere Transitive verbs (“loves”) everywhere. Existential quantification (“some”) everywhere	Lite-English, IS-A_1, IS-A_2, IS-A_3, IS-A_4
Intractable	Negation (“not”) in subject position Universal quantification (“only”) in subject position. Disjunction (“or”) in predate position	IS-A_5 IS-A_6 IS-A_7

that uses at most $\mathbf{O}(\log |\mathcal{D}|)$ space. This skectches a non-deterministic algorithm that decides KBQA using, overall, $\mathbf{O}(\log |\mathcal{D}|)$ space. ■

7. Conclusions

We have proposed to study the semantic complexity of controlled languages for managing OWL ontologies and ontology-based systems, such as ACE OWL and ACE OWL Lite (and similarly, of every controlled language that expresses OWL), which do not scale to data, by considering the data complexity of the description logics and the formal query language(s) these controlled languages and their fragments express. Our fine-grained analysis of the English constructs expressing all the possible combinations of description logic concepts (i.e., the concepts that may occur to the left and to the right of the concept inclusion symbol $\sqsubseteq$ in ontology assertions) shows that such controlled languages, which in general do not scale to data, contain, however, several significant fragments that scale when considered alongside GCQs and GCQ-English questions.

Table 6 provides a high-level summary of the data complexity associated to each of the combinations of constructs studied in declarative controlled English sentences. The upper row shows the maximal combinations of constructs that are tractable (i.e., in PTIME w.r.t. data complexity), while the lower row, shows the minimal combinations of constructs that are intractable (i.e., CONP-hard w.r.t. data complexity), viz., the constructs that when added, yield an exponential data complexity blowup. The use of relatively standard semantic (FO and λ-FO) meaning representations may indicate one reason for intractability: intractable combinations are “Boolean-closed”, giving rise to logics that can express (when combined with

(G)CQs) full Boolean complementation and intersection. If negation is constrained to occur only within predicate **VP** constituents (however complex), tractability ensues.

These results, which generalize those in [5, 19] and [18], Ch. 4., extend and refine the notion of semantic complexity proposed by Pratt and Third in [15] for English (controlled) fragments. As further work we would like to understand if further constraining English syntax, vocabulary, and semantics with the goal of achieving tractability, in the manner outlined by the tractable controlled languages defined in this paper, may negatively impact usability.

Aknowledgements. The authors wish to thank I. Pratt, R. Bernardi and N. Fuchs for their comments on the results presented here.

References

[1] ABITEBOUL, SERGE, RICHARD HULL, and VICTOR VIANU, *Foundations of Databases*, Addison-Welsey, 1995.

[2] ARTALE, ALESSANDRO, DIEGO CALVANESE, ROMAN KONTCHAKOV, and MICHAEL ZAKHARYASHEV, 'The *DL-Lite* family and relations', *Journal of Artificial Intelligence Research*, 36 (2009), 1–69.

[3] BAADER, FRANZ, DIEGO CALVANESE, DANIELE NARDI, PETER PATEL-SCHNEIDER, and DEBORAH MCGUINNESS, *The Description Logic Handbook*, Cambridge University Press, 2003.

[4] BARWISE, JON, and ROBIN COOPER, 'Generalized quantifiers and natural language', *Linguistics and Philosophy*, 4 (1980), 2, 159–219.

[5] BERNARDI, RAFFAELLA, DIEGO CALVANESE, and CAMILO THORNE, 'Lite natural language', in *Proc. of the 7th Int. Workshop on Computational Semantics (IWCS-7)*, 2007.

[6] CALVANESE, DIEGO, GIUSEPPE DE GIACOMO, DOMENICO LEMBO, MAURIZIO LENZERINI, and RICCARDO ROSATI, 'Data complexity of query answering in description logics', in *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006, pp. 260–270.

[7] EITER, THOMAS, GEORG GOTTLOB, EVGENY DANTSIN, and ANDREI VORONKOV, 'Complexity and expressive power of logic programming', *ACM Computing Surveys*, 33 (2001), 3, 374–425.

[8] FUCHS, NORBERT E., and KAAREL KALJURAND, 'Mapping Attempto Controlled English to OWL-DL', in *Demos and Posters of the 3rd European Semantic Web Conf. (ESWC 2006)*, 2006.

[9] HORROCKS, IAN, PETER F. PATEL-SCHNEIDER, and FRANK VAN HARMELEN, 'From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language', *J. on Web Semantics*, 1 (2003), 1, 7–26.

[10] JURAFSKY, DANIEL, and JAMES MARTIN, *Speech and Language Processing*, 2nd edn., Prentice Hall, 2009.

[11] KALJURAND, KAAREL, and NORBERT E. FUCHS, 'Bidirectional mapping between OWL DL and Attempto Controlled English', in *Proc. of the 4th Int. Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2006)*, 2006, pp. 179–189.

[12] KRISNADHI, ADILA, and CARSTEN LUTZ, 'Data complexity in the $\mathcal{EL}$ family of description logics', in *Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2007)*, 2007, pp. 333–347.

[13] MONTAGUE, RICHARD, 'Universal grammar', *Theoria*, 36 (1970), 3, 373–398.

[14] ORTIZ, MAGDALENA, DIEGO CALVANESE, and THOMAS EITER, 'Data complexity of query answering in expressive description logics', *Journal of Automated Reasoning*, 41 (2008), 1, 61–98.

[15] PRATT, IAN, and ALLAN THIRD, 'More fragments of language', *Notre Dame Journal of Formal Logic*, 47 (2006), 2, 151–177.

[16] SCHWITTER, ROLF, KAAREL KALJURAND, ANNE CREGAN, CATHERINE DOLBEAR, and GLEN HART, 'A comparison of three controlled natural languages for OWL 1.1', in *Proc. of the 4th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*, 2008.

[17] STAAB, STEFFEN, and RUDI STUDER, (eds.) *Handbook on Ontologies*, Int. Handbooks on Information Systems, Springer, 2004.

[18] THORNE, CAMILO, *Query Answering over Ontologies Using Controlled Natural Language*, Krdb dissertation series, Faculty of Computer Science, Free University of Bozen-Bolzano, 2010.

[19] THORNE, CAMILO, and DIEGO CALVANESE, 'Exploring Controlled English ontology-based data access', in *Proc. of the 2009 Workshop on Controlled Natural Language (CNL 2009)*, vol. 448 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/`, 2009.

[20] VARDI, MOSHE Y., 'The complexity of relational query languages', in *Proc. of the 14th Annual ACM Symposium on Theory of Computing*, 1982, pp. 137–146.

CAMILO THORNE
KRDB Research Centre for Knowledge and Data
Piazza Domenicani 3, 39100, Italy
`cthorne@inf.unibz.it`

DIEGO CALVANESE
KRDB Research Centre for Knowledge and Data
Piazza Domenicani 3, 39100, Italy
`calvanese@inf.unibz.it`