Query Answering over Ontologies Using Controlled Natural Languages

Query Answering over Ontologies Using Controlled Natural Languages

PhD Thesis in Computer Science

KRDB DS-2010-06

Camilo Thorne

Faculty advisor:

Prof. Diego Calvanese
Faculty of Computer Science
Free University of Bozen-Bolzano
Italy

Faculty co-advisor:

Dr. Raffaella Bernardi
Faculty of Computer Science
Free University of Bozen-Bolzano
Italy

Examination commitee:

Dr. Alessandro Artale, KRDB Research Centre, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy (secretary)
Dr. Giuseppe di Fabbrizio, AT&T Labs Research, Florham Park, NJ
Prof. Ralf Möller, School of Electrical Engineering and Informatics, TU Hamburg, Germany (chair)
Dr. Ian Pratt-Hartmann, Department of Computer Science, University of Manchester, UK

Date of public defense:

March 26, 2010

Printed and bound by DigiPrint.

ISBN: 978-8-890-50385-6

Contents

List of Figures

List of Tables

Acknowledgments

Scientia, aere perennius.
(Anonymous)

The mathematician and philosopher Blaise Pascal once said that minds drift between two kinds, between an *esprit de géométrie* and an *esprit de finesse*, bewteen analysis and intuition. Neither helped me, however, to forsee the long and in many ways unexpected path that lead me to Bolzano and to a PhD in logic and computer science. It was the motivation and passion communicated to me during many years by people coming from many different backgrounds that did. The list of aknowledgements risks accordingly to reach infinite proportions. I will try to be brief without omitting anybody.

I would like to thank first my supervisor, Diego Calvanese, for his patience, advice and wealth of personal qualities during these long years. Without his guidance through the jungle of logic and reductions I would have run astray. I would also like to thank my co-supervisor, Raffaella Bernardi, for introducing me to controlled languages and formal, computational semantics.

The criticisms to the draft of this thesis provided by Ian Pratt and Carlos Areces, external members of the PhD commitee, have also been invaluable, and so have been all the informal discussions and suggestions for further work that we exchanged during the past years, whether in Bolzano or in workshop venues. The discussions with Norbert Fuchs during his short visit to Bolzano, added to his skeptical remarks, contributed on the other hand greatly to better define the aims and the scope of my work. I must also mention the inspiration their work on (controlled or non-controlled) fragments of English and computational semantics gave me. Special thanks go to Werner Nutt, who introduced me to the theory of aggregate queries.

I would also like to thank all the people from the KRDB group, senior and junior, visiting and non-visiting, and specially fellow (or former) PhD students like Maria Keet, Mariano Rodriguez or Evgeny Kharlamov or Oscar Romero, which so much helped me to advance both professionally and personally. Working in the group has proved a wonderful experience of teamwork. To all, past, present and future, my gratitude.

I would also like to express my gratitude to those who sparked in me the taste and curiosity for mathematical logic. I would thus like to thank the people, mathematicians, philosophers and computer scientists alike, from my masters degree in Paris 1 and Paris 13 universities in Paris, and in particular my former MSc advisor from the LIPN, Denis Béchet, with whom my interest in the mathematics of natural languages started. Such interest would not had arised, however, without the people from my old Peruvian BA, the philosophy department of the PUCP, Lima. Nor can I forget the people from the LOA in Trento, or the charm of the KMi people in Milton Keynes.

Last, but not least, I would like to thank my parents, Carlos and Isabel Thorne, for instilling me with a curious mind. Or my friends, met in Peru, France or Italy, like César, Aline, Gustavo, Juan and Claudine, Rory and Jackie, to name but a few of those who colored the days and the nights that I luckily managed to spend in the outside world, despite so many deadlines.

Bolzano, November 2010.

Abstract

In this thesis we define and study the expressive power and the data complexity of a certain number controlled languages for ontology-based data access systems (OBDASs), in which data stored in relational databases is queried through an ontology (i.e., a "conceptual" or "intensional") middle layer and can be modeled by description logics.

Controlled languages and controlled language interfaces have been proposed as a means of enhancing the usability of human interfaces to information systems, and, in particular to OBDASs. Controlled languages are subsets of natural languages (such as English) with a limited vocabulary and syntax, designed to avoid the ambiguity inherent to arbitrary natural language utterances, that is very difficult to process.

These restrictions, however, do not address the impact the algebraic (a.k.a. as *semantic expressiveness*) and combinatorial/computational (a.k.a. as *semantic complexity*) properties the semantics of controlled languages might have on OBDASs. In particular, on the *scalability to data* of an OBDAS, viz., the computational complexity of query evaluation *in the size of the data*, a.k.a. the *data complexity* of query evaluation for OBDASs. Different combinations of controlled language function words (expressing different logical operations) may give rise to different computational properties.

We study this problem by proposing declarative and interrogative controlled languages that translate exactly and compositionally into (or *express*) different query and ontology languages, to single out combinations that are *(i) maximal w.r.t. tractable data complexity* (**PTime** or less) or *(ii) minimal w.r.t. intractable data complexity* (**NPTime**-hard or more), if not undecidable. We propose the following controlled languages. Lite-English, that expresses the description logic *DL-Lite*. DL-English, that expresses the description logic $\mathcal{ALCI}$. EL-English, that expresses the description logic $\mathcal{ELI}$. The IS-A$_i$s, which lie between Lite-English and DL-English. GCQ-English, that expresses tree-shaped queries. ATCQ-English, that expresses aggregate tree-shaped conjunctive queries.

We, moreover *(i)* propose a certain answers semantics (a generalization of the certain answers semantics for non-aggregate SQL queries over incomplete databases) for aggregate tree-shaped queries over OBDASs, and *(ii)* show that this semantics is a restriction of $\mathbf{HO}$ semantics to aggregate tree-shaped queries. We also analyze, by means of resolution decision procedures, the data complexity and expressiveness I. Pratt and A. Third's fragments of English.

It follows that, contrary to plain databases, where all these combinations taken together scale to data, any "Boolean closed" combination of controlled language constructors (function words) gives rise to intractable data complexity w.r.t. OBDAS query evaluation, while adding further restricted anaphoric pronouns may result in undecidability (and does when we consider arbitrary anaphoric pronouns).

Chapter 1

Introduction

1.1 Controlled Languages and their Scalability to Data

1.1.1 Natural Language Interfaces and the Usability Gap

Data is ubiquitous. Whether stored in relational databases or in knowledge bases the task of structuring, modelling, declaring, updating and querying data is a difficult one. In database management systems (DBMSs) the user executes these tasks using formal query languages, based often on formal logic, that combine both declarative and imperative features: for example the so-called Structured Query Language (SQL) or DATALOG (see [AHV95, EN04] for a general discussion on relational databases).

However, using these query and conceptual modelling languages requires previous training and can prove counterintuitive to the casual end-user. Database administration and data mining skills, together with domain expertise might be required, skills and expertise beyond those of casual end-users. Analogous problems have to be faced when dealing with knowledge bases [Sow99] or when "hybrid" systems are devised – that is, data management systems over which a reasoning layer, based on knowledge bases and ontologies, has been added. These drawbacks might be termed the *usability gap* in the management of databases and knowledge bases [KB07].

A proposal to bridge this usability gap are *natural language interfaces*, in which the casual end-user is allowed to type natural language questions, declarations and commands [ART95]. Such interfaces build a formal query making use of one of several language technologies. Studies have shown that in the context of data access, users indeed prefer natural language interaction, to, say, visual or formal query languages [KB07]. On the other, hand, as J. Sowa in [Sow99] accurately says, natural language is the ultimate knowledge representation language: we humans can depict naturally to our fellow human beings the world that surrounds us up to Cantor's paradise, to use a metaphor, and be understood.

Crucially, a natural language front end should map, by means of a mapping called *translation function*, natural language questions, declarations and orders to, resp., the formal queries, formal constraints and commands supported by the back-end, while preserving the semantics of those natural language expressions. To avoid tedious configuration and portability issues, it is customary to map natural language utterances first to an intermediate language, i.e., something like the *interlingua* or *pivot* of machine translation and only later, through drivers and bridges turn this into input for the DBMSs or information system [ART95]. Such approach implies several layers of processing, from the surface forms (the strings) to the syntax (the grammar) and the underlying semantics of the utterances (see Figure 1.1). Depending on how deep the translation method is, natural language interfaces can be classified into three main types:

– **Pattern-matching systems.** Queries are built via shallow parsing, using, e.g., regular expressions.

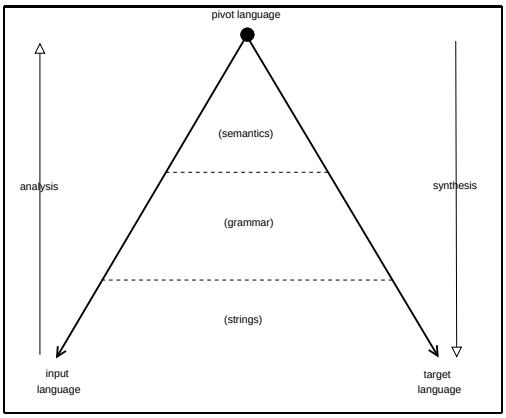


Figure 1.1: Machine translation.

- **Syntax-based systems.** Queries are built via (eventually probabilistic) parsing, using syntactic grammars such as unification grammars.
- **Semantic-based systems.** Queries are built via deep parsing using semantic (or dependency) grammars and in general grammars that generate semantic representations.

Many different natural language interfaces, mostly to DBMSs, have been proposed over the years. In general, the tendency has been to use deep translations combined with interlingua(s) based on formal logic. Among these we can mention the work by Clifford in [Cli88] in which a (symbolic) syntax and semantic-based system coupled with higher order logic ($\mathbf{HO}$) as pivot (or interlingua) and (temporal) SQL as target language is proposed, where $\mathbf{HO}$ is the logic obtained by enriching first order logic ($\mathbf{FO}$) with the types and constructors of the simply-typed lambda calculus [Car97]. STEP by Minock et al. [MON08, Min05] makes use of semantic grammars (domain-dependent but otherwise easy to define) and heuristics-based incremental deep semantic parsing coupled, again with $\mathbf{HO}$ as interlingua and SQL as target language. The ORAKEL system by Cimiano et al. [CHH+08] is geared instead towards semantic web knowledge bases rather than towards plain databases, and makes use of a (statistical) syntax-based system coupled with $\mathbf{HO}$, frame logic and an ontology layer. The system PRECISE by Popescu et al. [PEK03] proposes a pattern-matching system in which graph-theoretical algorithms are used to map a natural language question (seen as a sequence of words) to an SQL query. Similarly, in [DMB03] Dittenbach et al. propose a pattern-based natural language interface based on statistical machine translation. Finally, Giordani et al. propose in [Gio08, GM09] a syntax-based translation technique that predicts (via machine learning kernels that guess a SQL query plan, given a parse tree) the most likely SQL translation of a natural language question.

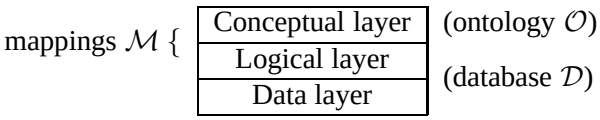
1.1.2 Ontology-Based Data Access Systems

In this thesis we are interested in one kind of information system, viz. in semantic web *ontology-based data access systems* (OBDASs). An *ontology* is, in general, a conceptualization of a domain of interest, providing the basic vocabulary and constraints holding over such a domain (see [Gua98] and [SS04], Chapter 1). They represent the intensional knowledge of the domain with a logical theory (i.e., a set of axioms) and can be of different kinds, e.g., foundational, domain, application ontologies, etc. Ontologies have been written in various formalisms, some of which are pictorial, like ER diagrams [Che76], UML class diagrams[1], conceptual graphs or semantic

[1] `http://www.uml.org/`

networks (see [BCN+03], Chapter 4). and others are based on notational, machine-processable variants of **FO** [CLN99, BCD05].

In the semantic web and OBDAS settings attention is restricted to OWL and description logic (description logic) ontologies. The World Wide Web Consortium standard for semantic web ontologies, the Web Ontology Language[2] (OWL) is formally underpinned by description logic ontology languages. An OBDAS is an information system in which an ontology is used as an interface or *conceptual view* to (possibly many) relational datasources accessed in the back-end of the system [CdL+05b]. Formally, it can be modelled as a triple $(\mathcal{O}, \mathcal{M}, \mathcal{D})$, where $\mathcal{O}$ is an OWL or description logic ontology (the conceptual layer), $\mathcal{D}$ a database (the logical layer and its physical implementation(s)) and $\mathcal{M}$ a set of mappings linking the concepts and relations of the ontology to $\mathcal{D}$'s relational schema, in a way similar to the Global-as-View perspective for database integration systems [Len02]:



OBDASs have been proposed as a specialization of semantic web ontologies and knowledge bases to data-intensive scenarios (e.g., accessing and integrating data from very large databases) [CdL+05b], where by a knowledge base we understand now a pair $(\mathcal{O}, \mathcal{D}_{\mathcal{M}})$, where $\mathcal{D}_{\mathcal{M}}$ is a *virtual database*, i.e., a set of relations populated modulo a set of mappings $\mathcal{M}$ (typically, SQL views). Such scenario can be regarded as an *incomplete information* setting, since databases in OBDASs specify only partially the knowledge of the domain. It is the ontology and its constraints or axioms that "completes" the knowledge of the domain by specifying the conditions any database that increases the factual information about the domain must comply with (see [SS04], Chapter 1). An OBDAS characterizes or represents a *class* of state of affairs, unlike databases that characterize a *single* state of affairs (i.e., they characterize completely the domain of interest). Queries, on the other hand, are SQL queries formulated over the vocabulary of the (top-level) ontology.

These intuitive notions can be given a formal meaning in **FO**. All OBDASs and knowledge bases can be seen, ultimately, as **FO** axiomatics and (core) SQL queries as syntactic sugar for **FO** formulas (see [AHV95], Chapter 8). Thus, the semantics of OBDASs can be captured by a **FO** *entailment* wherein we check whether queries are logically entailed by the OBDAS (or knowledge base). To be more precise, we are interested in retrieving the answers that hold in *all the logical models* of the system, a.k.a. as *certain answers* (see [CdL+05b, Len02] and [AHV95], Chapter 19).

1.1.3 Controlled Languages and the Ambiguity Problem

Since in OBDASs it is important to retrieve the exact set of answers of an information request, semantic- or syntax-based natural language interface making use of a deep and symbolic translation would be desirable. Building such an interface must tackle, however, the problem of natural language *ambiguity*: the same utterance may be parsed differently, or it can be ascribed different semantic representations. A proposal to overcome the problem of ambiguity in natural language interfaces for OBDASs are *controlled languages* and *controlled language interfaces* [BKGK05, KF06, SLH03, SKC+08, KB07]. A controlled language is a fragment of natural language (say, of English), with a limited lexicon and a small set of grammar rules [HSG04, MC99, Sow99]. Most importantly, controlled languages are engineered to strip them clean of ambiguity, so that their utterances "compile" into a unique ontology axiom and/or query, by restricting their syntax and their

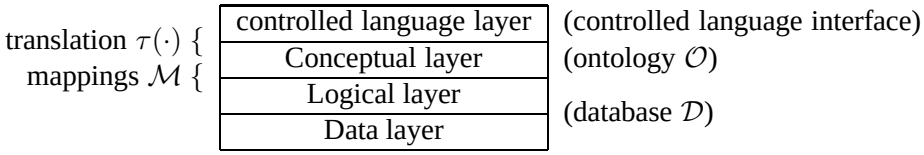
[2] `http://www.w3.org/TR/owl-ref/`

lexicon. They constitute a trade-off between the rigor of formal query and ontology languages and the intuitive appeal of natural language.

The *lexicon* of a controlled language is constituted by the set of its natural language words which, as in natural language, are partitioned into word *categories* (a.k.a. *parts-of-speech*), such as: common nouns ($\mathbf{N}$s, such as "car"), proper nouns ($\mathbf{Pn}$s, such as "John"), adjectives ($\mathbf{Adj}$s, such as "big"), intransitive, transitive and ditransitive verbs ($\mathbf{IV}$s, $\mathbf{TV}$s and $\mathbf{DTV}$s such as, resp. "runs", "eats" or "gives"), coordinators ($\mathbf{Crd}$s, such as "and", "or") pronouns ($\mathbf{Pro}$s, such as "it" or "that"), determiners ($\mathbf{Det}$s, such as "every") and prepositions ($\mathbf{Prep}$s, such as "to"), to name some. Words possess *morphological features* such as number, gender, tense, mood or voice, and combine recursively by means of a grammar (or set of syntax rules) into (potentially infinitely many) *constituents* and *sentences* or *utterances*. Such constituents can be classed into noun phrases ($\mathbf{NP}$s), verb phrases ($\mathbf{VP}$s), nominal compounds ($\mathbf{Nom}$s, $\mathbf{N}$s), relative clauses ($\mathbf{S}_g$s), prepositional phrases ($\mathbf{PP}$s), etc.

The meaning of controlled language utterances is *compositional*, i.e., a function of the meaning of their syntactic constituents. Such compositionality can be logically modelled by formal and computational semantics *compositional translations* $\tau(\cdot)$ that recursively map natural (and controlled) language utterances to $\mathbf{FO}$ and/or $\mathbf{HO}$ formulas known as *meaning representations* [Car97, Mon70, Moo97, HK98, Gam91, PWt93, BB05a], and which can be used as controlled language interface interlinguas.

Modulo $\tau(\cdot)$, we can *(i)* specify (or declare) the ontology (the domain constraints) with controlled language universally quantified declarative sentences S, *(ii)* specify and (or declare) information (or data) through controlled language facts F and *(iii)* formulate information requests (or queries) through controlled language questions Q. To retake the diagram we exhibited earlier, this amounts to adding a "natural language" layer to OBDASs:



The syntax of OWL, which is based on that of XML [SS04], is not meant for humans. Nor are description logics easy to manipulate for a user with no formal logic training. As an example consider the following English statement that affirms that for every human person, a parent exists

Every person has a father. (1.1)

In OWL, this mandatory participation of persons in the relation *hasParent*, holding among instances of *Person* and *Male*, would be written as:

```
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&;Person"/>
   <owl2xml:ObjectSomeValuesFrom>
     <owl2xml:ObjectProperty owl2xml:URI="&;hasParent"/>
     <owl2xml:Class owl2xml:URI="&;Male"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
```

and in description logic syntax as $\mathit{Person} \sqsubseteq \exists \mathit{hasParent}.\mathit{Male}$. Using a controlled language that translates unequivocally into description logic or OWL assertions overcomes the issue of human readability and understandability by non-logicians, preserving, at the same time, the properties of the (formal) ontology languages [KF06].

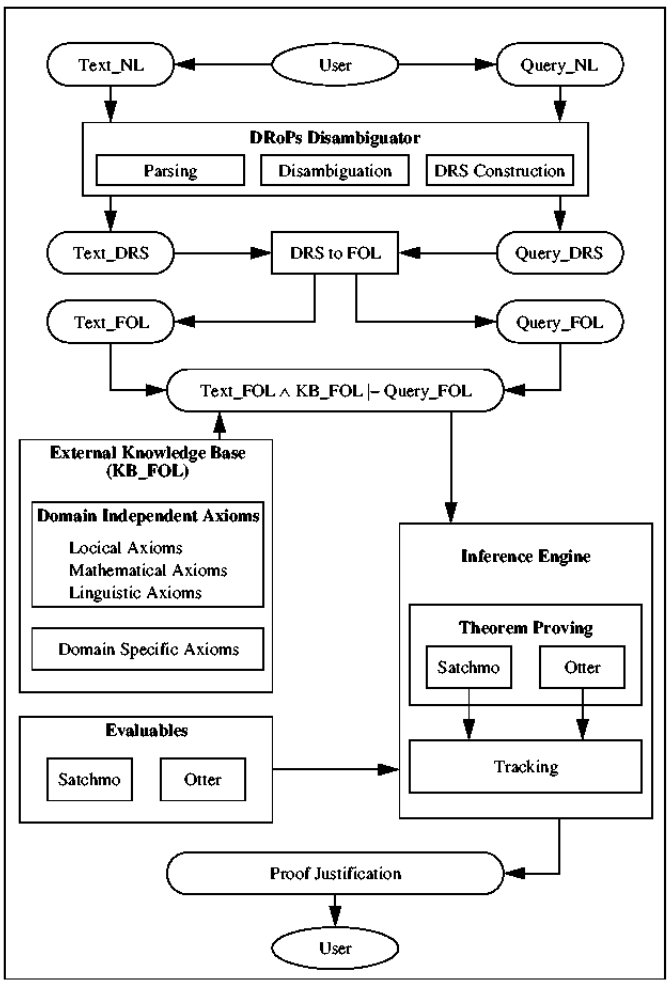


Figure 1.2: The architecture of RACE, the ACE Reasoning Engine, the ACE-based front end to knowledge bases [Sch05].

Controlled language	Compositional	Maps to	Parser	Goal
ACE [FKS05]	yes	FO	APE	Knowledge base authoring + querying
ACE-OWL [FK06]	yes	OWL-DL	APE	Ontology authoring + querying
PENG [SLH03]	yes	OWL-DL	ECOLE	Ontology authoring + querying
SOS [SKC$^+$08]	N.A.	OWL-DL	N.A.	Ontology authoring + querying
CLCE [Sow04]	yes	FOL	compiler	Knowledge representation
AECMA[Unw05]	no	no	no	User specifications
English Query [Blu99]	N.A.	SQL	N.A.	Database querying/management
OWL-CL language [ST06]	yes	OWL-DL	DCG parser	Ontology authoring
Easy English [Ber98]	no	no	no	User specifications
λ-SQL [MHWB06]	yes	SQL	compiler	Database querying
nRQL [Sch08]	yes	FO queries	DCG parser	Ontology querying
Rabbit [BBSS09]	no	OWL	GATE	Ontology authoring
ACE-PQL [BKGK05]	yes	PQL	DCG parser	Ontology authoring + querying
CLoNE [TBC$^+$07]	no	OWL	GATE	Ontology authoring
QE-III [Cli88]	yes	SQL	compiler	Database querying

Table 1.1: An overview of some controlled languages.

Most of the controlled language approaches to data management in the broad sense that exist are based on English. Of these, maybe the most interesting is Attempto Controlled English (ACE), a controlled language developed by Fuchs et al. at the university of Zürich [FS95, FKS05, FSH+05]. ACE is specifically devoted to knowledge representation tasks, i.e., declaring, updating, querying and reasoning over knowledge bases. In ACE, **FO** meaning representations are obtained modulo an interlingua of *discourse representation structures*, a notational variant of **FO** that copes with discourse anaphora [BB05b, KRv05]. What this means for ontology implementations is depicted by the architecture of the ACE Reasoning Engine (see Figure 1.2): ACE declarations (the "text" in the diagram) and questions (the "query" in the diagram) are parsed and translated into **FO**, then a **FO** theorem prover (Satchmo) or model builder (Otter), that checks whether they are consistent and/or entailed by a knowledge base, is invoked and the answers together with the proof (the "justification") are returned to the user.

Authoring the ontology of an ODBAS may also be seen as a data management task. As a result many controlled languages are geared towards OWL (and RDF) ontology authoring and, more in general, user specifications and constraint authoring, such as Schwitter's PENG (Processable English), Sowa's CLCE (Common Logic Controlled English), Rabbit or CLoNE, [BBSS09, TBC+07, BKGK05, ST06, SLH03, SKC+08]. ACE itself has a fragment, ACE-OWL, that can used to author OWL ontologies [Kal07, FK06, FKS05]. Controlled languages have been a topic of interest in industry as well as a means of writing down technical documents endowed with a common and unequivocal semantics: AECMA Simplified English [Unw05], for instance, has been proposed for standardizing official document translation within the European aerospace industry. Controlled language interfaces have also been proposed as DBMS front-ends: an example of an industrially developed controlled language for databases is Microsoft's English Query, which adds natural language support to Microsoft SQL servers [Blu99] or λ-SQL by Winter et al. See Table 1.1 for a summary.

1.1.4 Semantic Complexity and the Scalability Problem

If controlled languages and controlled language interfaces provide an answer to the problem of ambiguity (thereby bridging the usability gap) and, furthermore, provide an answer that is tractable w.r.t. to natural language processing (compositional translations $\tau(\cdot)$ can be computed in time polynomial in the size of the input controlled language utterances), we must also consider their *semantic complexity*. By the semantic complexity of a controlled language, or, more in general, any fragment of a natural language such as English, we understand, following Pratt in [PH08a] the *computational complexity of the logic reasoning decision problems that apply to their meaning representations*. One such problem is the satisfiability (SAT) problem of a set of natural language utterances, wherein we check whether their meaning representations contain no inconsistencies.

Semantic complexity may affect the performace of an ontology authoring tool insofar as these tools are constantly checking for the satisfiability (consistency) of the ontologies authored [SKC+08]. This observation holds also for controlled language interfaces to OBDASs. Crucially, query and ontology language constructs affect the *data complexity* of data management tasks, i.e., the computational complexity of OBDAS reasoning and query evaluation measured w.r.t. the *size of the data* [Var82]. Data complexity provides a measure of the *scalability to data* of an OBDAS or a relational database: tractable data complexity implies scalability, while intractable data complexity, i.e., exponential in the data, precludes (good) scalability (see [CdL+06, Var82, OCE08, GHLS07], together with [AHV95], Chapter 16, and [SS04], Chapter 1).

Now, semantics (and formal semantics) allows us to group the categories of controlled language lexicons into two main classes:

– **Content words.** Content words are words that denote individuals, sets and relations, viz.,

resp., **Pn**s (individuals), **N**s, **Adj**s, **IV**s (sets), **TV**s and **DTV**s (binary and ternary relations). It is said to be an open class since nothing prevents a natural language or a controlled language from having arbitrarily many (see [JM09], Chapter 5).

– **Function words.** Function words are words that denote functions among those individuals, sets and relations. They basically belong to the **Det**, **Pro**, **Crd**, **Relp** and **Prep** categories and constitute a closed class (see [JM09], Chapter 5).

In particular, function words map modulo $\tau(\cdot)$ into **Fo** constructors (**Det**s and indeterminate **Pro**s map into quantifiers, **Crd**s map into logical conjunction and disjunction, and so forth). This means that the coverage, and especially the function words (or, by abuse, constructs) present in the controlled language will affect modulo $\tau(\cdot)$ the performance of OBDASs. Using a pair of declarative and interrogative controlled languages covering a combination of *tractable* constructors will give rise to scalable systems, whereas an *intractable* combination will have the opposite effect.

Results regarding controlled languages and natural language fragments w.r.t. SAT (using computational semantics techniques) can be found already in the literature. Slavkovic in [Sla07] and Pratt et al. in [PH01, PHT06, PH04, PHM09, PH08a] provide both lower and upper complexity bounds for SAT. In the case of [Sla07], upper bounds (by means of answer set programming) are given for fragments and controlled languages that map into the two variable fragment of **Fo**. In the case of Pratt et al. a family of *fragments of English*, powerful enough to capture commonsense syllogistic reasoning, is proposed and tight complexity bounds (by means of resolution automated theorem proving procedures) are shown. Full ACE is known to be undecidable for SAT [FK06, FKS05], whereas ACE-OWL is decidable for SAT, which follows from the decidability of the description logic it expresses, $\mathcal{SROIQ}$ [HKS06].

1.2 Aims and Methodology of this Thesis

1.2.1 Aims and Goals

In this thesis we pursue two goals. On the one hand, we want to study the *expressive power* of controlled languages. On the other hand, their *semantic complexity*, with particular emphasis on data complexity. We believe that expressive power and data complexity provide good tools for understanding the scalability to data of controlled languages for OBDASs.

As we have seen, controlled language interfaces to OBDASs assume as source language a controlled language, which is to be translated by a rule-based, symbolic and compositional syntax-directed translation algorithm (in a way similar to programming languages' compilation [AUS86]) into a formal query and/or ontology language. In particular, this translation algorithm computes a so-called formal semantics compositional translation $\tau(\cdot)$ and is sound and complete w.r.t. such translation $\tau(\cdot)$, thus ensuring complete accuracy. Moreover, such computation is polynomial in the input strings and independent from the data for the purposes of data access. This is important since query evaluation algorithms for the back-end OBDAS should be sound and complete and therefore not subject to precision and/or recall information theoretic metrics.

However, under these requirements and assumptions, different computational properties for query evaluation may arise with different choices of natural language constructors, and this may penalize the scalability of the controlled language interface. To understand how and why this may happen, we focus on two of the main OBDAS management tasks, namely *(i)* query evaluation, which we model through the knowledge base *query answering* decision problem (KBQA), and *(ii)* knowledge and data specification, which we model through the knowledge base *consistency* decision problem (KBSAT). A fine-grained analysis of the computational properties of these decision (or reasoning) problems involves analyzing the interaction of data and constraints, by possibly fixing (or "parameterizing") some of their inputs:

– **data complexity:** we want to know their computational complexity when we consider the data as their only input, while the ontology and the query are considered fixed.
– **combined complexity:** we want to know their computational complexity when we consider all their inputs (database, ontology and query).

The data complexity analysis is the most relevant measure in the setting we consider: in relational database and/or OBDAS settings the size of the data (measured in terms of the tuples and/or individuals in the database) will outsize, by and large, the size of the queries and/or that of the ontology.

1.2.2 Methodology and Scope

The methodology we propose is to *express* KBSAT and KBQA in controlled language. By this we mean three things: *(i)* define a declarative controlled language, *(ii)* define an interrogative controlled language and *(iii)* define compositional translations mapping the former to a formal ontology language and the latter to a formal query language. This done, consider the resulting formal query and ontology languages and study both their (relative and absolute) expressive power and their computational properties. This is to be done, moreover, "piece-meal" or "incrementally", i.e., construct-wise, so that we can study the properties of the controlled language constructs in isolation by answering in particular the following questions:

– which combinations of controlled language constructs in declarative sentences and questions give way to *maximal* (w.r.t. expressive power) tractable (i.e., **PTime** or less) data complexity for either KBSAT or KBQA, and
– which combinations of controlled language constructs in declarative sentences and questions give way to *minimal* (w.r.t. expressive power) intractable (i.e., at least **NPTime** or **coNPTime**-hard) data complexity for either KBSAT or KBQA?
– which is the expressiveness (i.e., the model theoretic properties) of the controlled languages defined in this manner?

In so doing we rely on and extend results coming from both the formal semantics and ontology literature. Compositional translations are defined exploiting all the possibilities set by $\mathbf{Ho}$ typing (see [Moo97, Car97, Mon70, PWt93, HK98, Gam91]) and by the theory of generalized quantifiers (see [Mon73, BC80]) to express complex ontology languages and query languages containing, possibly, SQL aggregation functions (which require a so-called bag-set semantics).

Regarding in particular questions, we extend the work of Karttunen [Kar77] and Clifford [Cli88] by considering more expressive classes of questions (aggregate questions), queries (SQL aggregate queries) and $\mathbf{Ho}$ meaning representations. Regarding semantic complexity, we generalize the work on the (syllogistic) Fragments of English by Pratt & Third [PHT06] by studying the computational properties (the data complexity) of the $\mathbf{Fo}$ characterizations of the semantics of OBDASs as stated by Calvanese et al. [CdL$^+$06], thus restricting our attention to description logic-based ontologies.

1.3 Structure of this Thesis

Chapters 2 and **3** provide the basic notions and notation of this thesis. **Chapter 2** recalls formal semantic theory, compositional translations and semantically enriched grammars (the formalism we mainly use for defining controlled languages) and the ensuing notion of semantic complexity. Pratt and Third's fragments of English are also recalled. **Chapter 3** introduces description logic-based ontology languages, knowledge bases, conjunctive queries (the formal counterpart of

SELECT-PROJECT-JOIN SQL queries), tree-shaped conjunctive queries and, specifically, the family of *DL-Lite* description logics, whose computational properties we also outline. We propose to derive computational properties by *expressing* in controlled language formal ontology and query languages and their decision problems.

Chapter 4 is devoted to the definition of the declarative controlled language Lite-English, expressing *DL-Lite*, and of the interrogative controlled language GCQ-English, expressing tree-shaped conjunctive queries, to their expressiveness and semantic complexity. In particular, we compare the expressiveness of Lite-English to that of Pratt and Thirds fragments of English.

In **Chapter 5** we express aggregate tree shaped queries in controlled language by means of the controlled language ATQ-English. Such queries are syntactic sugar for SQL aggregate queries. We propose a certain answers semantics for answering aggregate tree shaped queries over ontologies, OBDASs and knowledge bases. We show that $\mathbf{HO}$ meaning representations capture such queries exactly and that $\mathbf{HO}$ is powerful enough to capture their certain answers semantics. We then proceed to extend the coverage to full negation, universal restrictions and comparisons/comparatives (not covered by tree shaped queries) and look at data complexity. We show that aggregations do not significantly increase data complexity, but that comparisons, full negation and universal restrictions turn data complexity hard for **coNPTime**.

In **Chapter 6** we study a family of controlled languages that express description logics that lie between *DL-Lite* and the description logic $\mathcal{ALCI}$ (the least description logic closed under boolean operations containing *DL-Lite*): the IS-A_is, EL-English and description logic-English and study their data complexity for query answering. This with the purpose of defining controlled languages that are maximal w.r.t. tractable data complexity and minimal w.r.t. intractable data complexity.

In **Chapter 7** we study the data and combined complexity of query evaluation and consistency checking for Pratt's fragments of English. We also strengthen Pratt's undecidability results to those fragments and questions covering (restricted) anaphoric pronouns.

Chapter 8 summarizes the results of the thesis and outlines the possible directions in which this research can be extended.

Chapter 2

Compositionality and Semantic Complexity

In this chapter we give an overview of formal and computational semantics for English, of some standard methods (viz., semantically enriched grammars) used to generate such formal semantics and of the crucial issue of the *semantic complexity* of English (introduced by I. Pratt and Allan Third in [PHT06]) that ensues. Intuitively, to achieve a formal semantics for English, we need to use formal logic(s), namely first order logic (**Fo**) and higher order logic (**Ho**) as "glue", which, modulo a compositional translation, recursively put in correspondence natural language surface forms to set-theoretical denotations. Such translations can be easily defined using semantically enriched grammars.

The logic expressions associated to natural language utterances by compositional translations are known in the literature as *meaning representations*. Many natural language complete utterances possess **Fo** meaning representations. Semantic complexity describes the computational properties of the decision reasoning problems to which **Fo** and **Ho** natural language meaning representations give rise.

Such semantic model makes in particular clear that English function words (**Det**s, **Relp**s, etc.) denote logical operators and content words (**N**s, **TV**s, etc.), n-ary relations and individuals (of some domain of interest). We also introduce a certain number of formal semantic notions and notation that will be used throughout the remainder of this thesis. We finish by briefly recalling Pratt and Third's important family of fragments of English, to which the notion of semantic complexity was first applied.

The sections on the typed-lambda calculus, **Ho** and compositionality are based on Chapters 2 and 3 of [Car97]. The standard **Ho** meaning representations for natural language constituents are also derived mainly from Chapter 3 of [Car97]. Regarding semantically enriched grammars, we adapt [JM09], Chapter 18. Finally, the section on semantic complexity is an elaboration of [PHT06] and (to a lesser extent) [PH08a].

2.1 Formal And Computational Semantics

2.1.1 First Order Logic and Higher Order Logic

As is customary in the literature, we introduce **Ho** as the restriction of a more powerful system, the simply-typed lambda calculus. Thereafter, **Fo** can be defined as a proper fragment of **Ho**. Proceeding in this manner has its advantages: although both **Ho** and its fragment **Fo** are undecidable, for **Fo** sound and complete deductive calculi exist, i.e., we can provide combinatorial characterizations of what it means to derive truths from truths.

Let $\mathbf{C} := \{c_i \mid i \in \mathbb{N}\}$ be a a countably infinite set of *constants* and $\mathbf{V} := \{x_i \mid i \in \mathbb{N}\}$ a countably infinite set of *variables*. The set *Exp* of *expressions* or *terms* u of the *simply typed*

lambda calculus is defined by the grammar

$$u \to c_i \mid x_i \mid u(u') \mid \lambda x_i.u.$$

Given a term u, the set $FV(u)$ of its free variables is defined by induction on u as follows: *(i)* $FV(c) := \emptyset$, *(ii)* $FV(x) := \{x\}$, *(iii)* $FV(u(u')) := FV(u) \cup FV(u')$ and *(iv)* $FV(\lambda x.u) := FV(u) \setminus \{x\}$.

Let $\mathbf{B} := \{t_i \mid i \in \mathbb{N}\}$ be a countably infinite set of *basic types*. The set *ST* of *simple types* T is defined by the grammar

$$T \to t_i \mid T \to T.$$

A *typing* is a function $\chi\colon Exp \to ST$. Whenever $\chi(u) = T$ we write $u{:}T$, and call it a *declaration*. A *context* is a finite set $E := \{x_1{:}T_1, \ldots, x_n{:}T_n\}$ of variable declarations. We say that an expression u is *typable* with type T w.r.t. context E and write $E \vdash u{:}T$ whenever $x{:}T \in \Gamma$, for every $x \in FV(u)$. A *typing rule* is any of the following:

$$dv\ \frac{}{E, x{:}T \vdash x{:}T}$$

$$app\ \frac{E \vdash u{:}T \qquad E \vdash u'{:}T \to T'}{E, E' \vdash u'(u){:}T'} \qquad abs\ \frac{E, x{:}T \vdash u{:}T'}{E \vdash \lambda x^T.u{:}T \to T'}$$

A *type judgement* is a finite tree rooted on an expression u typable with type T w.r.t. a context Γ and recursively generated using the typing rules. An expression u is said to be *well-typed* of type T iff there exists a type judgment rooted on $E \vdash u{:}T$ with $E = \emptyset$ [Lal97].

A *substitution* is a partial mapping $\sigma\colon \mathbf{V} \to Exp$, consistent with typing in the sense that $\sigma(x) := u$ iff x and u are typed identically. Substitutions can be inductively extended to a mapping over arbitrary terms in the usual way. We denote by $u\sigma$ the result of applying σ to term u. As is common in the literature, we denote substitutions by sets $\{x_1 \mapsto u_1, \ldots, x_n \mapsto t_n\}$ of associations mapping the variable x_i to the expression u_i, for $i \in [1, n]$.

We say that a term u *reduces in a single step* to a term v, denoted $\rhd_1$, iff $u = \lambda x.u'(u'')$ and $v = u''\{x \mapsto u'\}$. The relation denoted $\rhd$, called *beta reduction*, is the reflexive and transitive closure of reduction in one step.

A term u is said to be a *redex* if it is of the form $u = \lambda x.u'(u'')$. Otherwise it is called a *reduct*. A term u is said to be *(strongly) normalizable* iff there exits a reduct v and a finite sequence of reductions s.t. $u \rhd u_1 \rhd \ldots \rhd u_n \rhd v$, in which case v is called the *normal form* of u. Simply-typed lambda calculus well-typed expressions are strongly normalizable. Furthermore, the normal form is unique and normalization order-independent.

Definition 2.1.1 (Higher Order Logic)**.** The system of $\mathbf{HO}$ is obtained by restricting the set $\mathbf{Bas}$ of basic types of the simply typed lambda calculus to $\{e, t\}$, where e stands for the type of individuals and t for that of Booleans (truth values).

A $\mathbf{HO}$ *formula* is an expression φ of type t and a *sentence* a formula s.t. $FV(\varphi) = \emptyset$. Notice that we use Greek letters, φ, ψ, etc. to denote formulas and sentences. Formulas and sentences are, in particular, built using the $\mathbf{HO}$ constants $\wedge{:}t{\to}t$, $\neg{:}t{\to}t$, $some{:}(T{\to}t) \to t$ and $\approx{:}T{\to}(T{\to}t)$, and by setting, for φ and ψ of the convenient type, $\exists x^T\varphi := some_T(\lambda x^T.\varphi)$, $\forall x^T\varphi := \neg\exists x^T\neg\varphi$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \Rightarrow \psi := \neg\varphi \vee \psi$ and $\varphi \Leftrightarrow \psi := (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$.

The semantics of the simply typed lambda calculus and a fortiori that of $\mathbf{HO}$ is given by mapping type-theoretical expressions to denotational frames. We adopt the convention of writing type $T_1 \to (\ldots (T_n \to T) \ldots)$ as $T_1 \times \cdots \times T_n \to T$.

Let $\{\mathbf{D}_{t_i}\}_{i\in\mathbb{N}}$ be a family of *basic domains* (i.e., one for every basic type t_i). Denotational *frames* $\mathbf{Dom}$ are defined by structural recursion on types as follows:

$$\begin{aligned}\mathbf{Dom}_{t_i} &:= \mathbf{D}_{t_i},\\ \mathbf{Dom}_{T\to T'} &:= \mathbf{Dom}_{T'}^{\mathbf{Dom}_T},\\ \mathbf{Dom} &:= \bigcup_{T\in ST} \mathbf{Dom}_T.\end{aligned}$$

An *interpretation function* is a mapping $\cdot^{\mathcal{I}}\colon \mathbf{C}\to\mathbf{Dom}$ s.t. $c^{\mathcal{I}}\in\mathbf{Dom}_T$ iff c is a constant of type T. An *interpretation* is a tuple $\mathcal{I} := (\mathbb{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$ where *(i)* $\mathbb{D}_{\mathcal{I}}\subseteq\mathbf{Dom}$ and *(ii)* $\cdot^{\mathcal{I}}$ is an interpretation function. An *assignment* is a function $\gamma\colon \mathbf{V}\to\mathbf{Dom}$ such that $\gamma(x)\in\mathbf{Dom}_T$ iff x is a variable of type T.

Definition 2.1.2 (Denotation)**.** The *denotation* $u^{\mathcal{I}}_{\gamma}$ of an expression u w.r.t. γ and $\mathcal{I}$ is defined by recursion on expressions as follows:

- $x^{\mathcal{I}}_{\gamma} := \gamma(x)$,
- $c^{\mathcal{I}}_{\gamma} := c^{\mathcal{I}}$,
- $u(v)^{\mathcal{I}}_{\gamma} := u^{\mathcal{I}}_{\gamma}(v^{\mathcal{I}}_{\gamma})$ and
- $\lambda x.u^{\mathcal{I}}_{\gamma} :=$ the function f s.t. $f(c) = u^{\mathcal{I}}_{\gamma[x:=c]}$,

where $\gamma[x := c]$ is the assignment identical to γ.

Definition 2.1.3 (Satisfaction)**.** We say that an interpretation $\mathcal{I}$ *satisfies* a $\mathbf{Ho}$ formula φ w.r.t. assignment γ whenever the following conditions hold:

- $\mathcal{I},\gamma\models\varphi$ iff $\varphi^{\mathcal{I}}_{\gamma}=1$,
- $\mathcal{I},\gamma\models\neg\varphi$ iff $\mathcal{I},\gamma\not\models\varphi$,
- $\mathcal{I},\gamma\models\varphi\wedge\psi$ iff $\mathcal{I},\gamma\models\varphi$ and $\mathcal{I},\gamma\models\psi$, and
- $\mathcal{I},\gamma\models\exists x^T\varphi$ iff there exists some $\gamma'[x := d]$ s.t. $\mathcal{I},\gamma'[x := d]\models\varphi$.

We say that $\mathcal{I}$ is a *model* of formula φ, written $\mathcal{I}\models\varphi$, whenever for all γ, $\mathcal{I},\gamma\models\varphi$. For every set of formulas Γ, we write $\mathcal{I}\models\Gamma$ if, for all $\varphi\in\Gamma$, $\mathcal{I}\models\varphi$. We denote *Mod*$(\varphi)$ (resp. *Mod*(Γ)) the (class of) models of φ (resp. Γ). We say that ψ (resp. Γ) *entails* φ, written $\psi\models\varphi$ (resp. $\Gamma\models\varphi$), whenever *Mod*$(\psi)\subseteq$ *Mod*(φ) (resp. *Mod*$(\Gamma)\subseteq$ *Mod*(φ)). When the converse also holds, we say that φ and ψ are *equivalent* and write $\varphi\equiv\psi$.

Definition 2.1.4 (First Order Logic)**.** The system of $\mathbf{Fo}$ can be seen as a subset of $\mathbf{Ho}$, where formulas are built from the following basic syntactic constructs: *(i)* individual variables x of type e, *(ii)* individual constants c of type e, *(iii)* function constants f of arity n and type $e\times\cdots\times e\to e$ and *(iv)* predicate constants R of arity n and type $e\times\cdots\times e\to t$.

The set of $\mathbf{Fo}$ terms t is built in the usual way by recursively combining individual constants, individual variables and function constants together (see [Car97], Chapter 3). A $\mathbf{Fo}$ *signature* is a triple $\mathbf{Sig} := (\mathbf{C},\mathbf{F},\mathbf{R})$, where $\mathbf{C}$ is a set of individual constants, $\mathbf{F}$ a set of function constants (a.k.a. function symbols) and $\mathbf{R}$ a set of predicate/relation constants (a.k.a. predicate/relation symbols).

Remark 2.1.5. Since predicate symbols in $\mathbf{Fo}$ denote *characteristic functions* and can be seen as denoting *set-valued relations*, the domain is accordingly restricted to $\mathbf{Dom} := \mathbf{D}_e$. Similarly, interpretations $\mathcal{I}$ map: *(i)* individual constants c to points $c^{\mathcal{I}}\in\mathbb{D}_{\mathcal{I}}$, *(ii)* function constants f of arity n to functions, $f^{\mathcal{I}}\colon \mathbb{D}^n_{\mathcal{I}}\to\mathbb{D}_{\mathcal{I}}$ and *(iii)* predicate constants S of arity n to subsets $S^{\mathcal{I}}\subseteq\mathbb{D}^n_{\mathcal{I}}$. Assignments map variables to elements in $\mathbf{Dom} := \mathbf{D}_e$. Notice, finally, that now $\mathcal{I},\gamma\models S(\bar{t})$ whenever $\gamma(\bar{t})\in S^{\mathcal{I}}$.

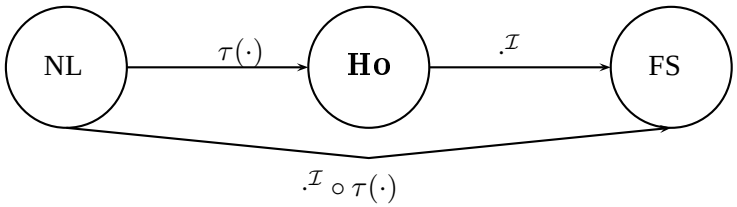


Figure 2.1: The homomorphism principle and the syntax-semantics interface between natural language syntax (NL) and formal semantics (FS).

2.1.2 Compositional Translations and Grammars

Following the work laid out by logicians such as R. Montague [Mon70, Mon73], there is a broad consensus that *(i)* the notion of meaning relies on the notion of denotation and truth (i.e., truth conditions) *(ii)* meaning exhibits a predicate-argument (or functional) structure *(iii)* meaning is *compositional*, i.e., satisfies the *compositionality* (**Com**) principle:

The meaning of a sentence is a function of the meaning of its constituents. (**Com**)

The so-called *syntax-semantics interface* aims at studying the relationships that exist between natural language syntax and natural language semantics. Montague [Mon70, Mon73] and Lambek [Lam58], and later Moortgat in [Moo97], showed that the syntax-semantics interface can be realized and conditions *(i)* – *(iii)* satisfied by adopting the strategy depicted in Figure 2.1.

Such a strategy consists in defining a homomorphic mapping $\tau(\cdot)$, known in the literature as a *compositional translation*, between a fragment (or controlled fragment) of a natural language like English and formal logic, in particular higher order logic ($\mathbf{Ho}$). Modulo this mapping $\tau(\cdot)$ one can assign $\mathbf{Ho}$ denotations to language constituents by composing $\tau(\cdot)$ with $\cdot^{\mathcal{I}}$. Actually, Montague used a system that extends $\mathbf{Ho}$ with modalities, intensional logic, which can be used to capture, compositionally, e.g., English moods and tense [Mon70, Mon73]. Given that we do not consider modalities or tense in the controlled languages studied in this thesis, we will not consider such extension either.

Compositional Translations. Let $\mathbf{\Sigma}$ denote an *alphabet* or set of basic symbols, and $\mathbf{\Sigma}^*$ its Kleene *star*, the set of all finite strings (sequences of symbols) i.e., the closure of $\mathbf{\Sigma}$ under the string $\cdot$ concatenation operator. A *language* is any subset of $\mathbf{\Sigma}^*$. By $\mathbf{\Sigma}^+$ we denote the positive star of $\mathbf{\Sigma}$, i.e., $\mathbf{\Sigma}^* \setminus \{\epsilon\}$, where ϵ denotes the empty string (see [UHM01], Chapter 1).

Definition 2.1.6 (Compositional Translation)**.** Let L be a *source* language over an alphabet $\mathbf{\Sigma}$ (i.e., $L \subseteq \mathbf{\Sigma}^*$), and L' a *target* language over an alphabet $\mathbf{\Sigma}'$ (i.e., $L' \subseteq \mathbf{\Sigma}'^*$). A *compositional translation* from L to L' is a function $\tau(\cdot)$ from $\mathbf{\Sigma}^*$ to $\mathbf{\Sigma}'^*$ such that the following conditions hold:

- for all $w_1 \cdots w_n \in L$, $\tau(w_1 \cdots w_n) = \tau(w_{\pi(1)}) \cdots \tau(w_{\pi(n)})$, where $\pi(\cdot)$ is a permutation over $\{1, \ldots, n\}$, and
- for all $w \in \mathbf{\Sigma}^*$, $w \in L$ iff $\tau(w) \in L'$.

For every set $W \subseteq L$, we define $\tau(W) := \{\tau(w) \mid w \in W\}$. Note that permutations are needed since word order in the source language may not necessarily reflect word order in the target language.

Semantically Enriched Grammars. Given a fragment (controlled or otherwise) of natural language, we can build grammars which define both the language utterances and the compositional translation $\tau(\cdot)$. Many classes of grammars can be used to this purpose. A well-known class is the class of categorial grammars, also called logical or type-theoretical grammars [Car97, Moo97, Lam58]. Another, more simple, class, based on the context-free grammar model, is the class of grammars with semantic actions (see [JM09], Chapter 18). Grammars with semantic actions are context-free grammars that have been put in correspondence which a set of semantic actions that define $\tau(\cdot)$.

Definition 2.1.7. A *semantically enriched grammar* is a context-free grammar of the form $G := (\mathbf{\Sigma}, \mathbf{Cat}, \mathbf{Lex}, \mathbf{Rul}, \mathbf{S}, \tau(\cdot))$ where

- $\mathbf{\Sigma}$ is an alphabet, a.k.a. set of *words*,
- $\mathbf{Cat}$ is a set of symbols called *categories*,
- $\mathbf{Lex} \subseteq \mathbf{Cat} \times \mathbf{\Sigma}$ is a *lexicon*,
- $\mathbf{Rul} \subseteq \mathbf{Cat} \times (\mathbf{\Sigma} \cup \mathbf{Cat})^+$ is a set of *phrase structure rules*,
- $\mathbf{S} \in \mathbf{Cat}$ is a distinguished category called the *terminal* category, and
- $\tau(\cdot)$ is a $\mathbf{Ho}$ compositional translation.

Any sequence $w \in (\mathbf{\Sigma} \times \mathbf{Cat})^*$ is called a *constituent*. If $(C, w) \in \mathbf{Lex}$ or $(C, w) \in \mathbf{Rul}$ we write $C \to w$. Moreover, as customary in the literature, we partition $\mathbf{Cat}$ into two sets, viz., a set of *basic* categories and a set of *non-basic* categories. Basic categories are those that occur in the lexicon, and correspond to the parts of speech: the common noun ($\mathbf{N}$), the adjective ($\mathbf{Adj}$), the relative pronoun ($\mathbf{Relp}$), the determiner ($\mathbf{Det}$), etc. Categories can be, furthermore, *subcategorized* , i.e., multiplied so as to model (morphological) agreement in person number, gender, tense, polarity, mood, voice, etc. (e.g., $\mathbf{N}^m_{pl}$ would stand for a masculine plural (common) noun $\mathbf{N}$).

By exploiting the phrase structure rules and lexicon of a grammar G, $\tau(\cdot)$ can be defined by means of *semantic actions*:

- for each $C \to w \in \mathbf{Lex}$, we specify $\tau(C)$, and
- for each $C \to C_1 \cdots C_n \in \mathbf{Rul}$, we write $\tau(C) := \tau(C_{\pi(1)})(\dots \tau(C_{\pi(n)}) \dots)$.

This means that $\tau(\cdot)$ will be recursively defined on syntactic constituents, i.e., if w is a word from the alphabet, then $\tau(w)$ is a $\mathbf{Ho}$ expression and if $w = w_1 \cdots w_n$, then $\tau(w) := \tau(w_{\pi(1)})(\dots \tau(w_{\pi(n)}) \dots)$ (or, to be more, precise, its β-reduct). Notice that semantically enriched context-free grammars are, essentially, the context-free grammars with semantic actions of compiler theory (see, e.g., [AUS86], Chapter 4).

We say that G *derives in one step* a constituent w from a category C, written $C \Longrightarrow_G w$ whenever $C \to w \in \mathbf{Lex}$ and $\tau(C)$ is a well-typed $\mathbf{Ho}$ formula. The *derives* relation $\Longrightarrow^*_G$ is then defined as the reflexive and transitive closure of $\Longrightarrow_G$. If $C \Longrightarrow^*_G w$, we say that there is a derivation of w *rooted* in category C. The *generated language* of G is then defined as $L(G) := \{w \in \mathbf{\Sigma}^* \mid \mathbf{S} \Longrightarrow^*_G w\}$. The index G can be omitted whenever the grammar is clear by context.

The logic fragment *expressed* by G and/or $L(G)$ is the $\mathbf{Ho}$ fragment $\mathbf{L}_{L(G)} := \{\tau(w) \mid w \in L\} = \tau(L(G))$. Every meaning representation $\tau(w) \in \mathbf{L}_{L(G)}$ is said to be *expressed* by G and/or $L(G)$.

As it is typical of context-free grammars, derivations can be captured by parse trees. Once a parse tree is computed, the compositional translation $\tau(\cdot)$ can be trivially computed bottom-up from leaves to root by applying siblings to each other, normalizing and checking well-typedness. The computation can be done on the fly. Parsing and semantic evaluation take time polynomial in the length $|w|$ of an input string w.

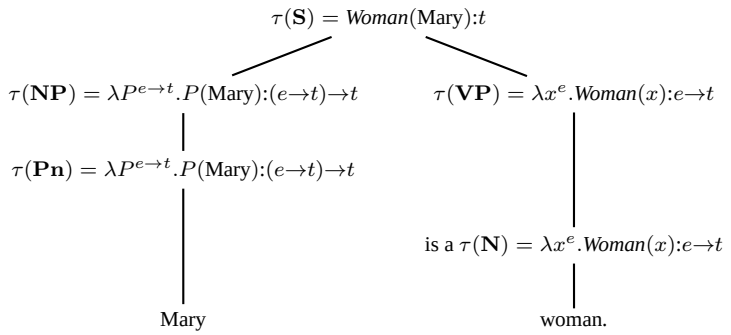


Figure 2.2: Parse tree for the COP sentence "Mary is a woman.".

Meaning Representations, Content and Function Lexicons. The $\mathbf{Ho}$-valued compositional translation $\tau(\cdot)$ defined by a semantically enriched grammar associates to each of the constituents of the language it generates $\mathbf{Ho}$ expressions called *meaning representations*. A $\mathbf{Ho}$ fragment $\mathbf{L}$ and a $\mathbf{Ho}$ formula φ are said to be *expressible* by a controlled or non controlled fragment L of natural language whenever a semantically enriched grammar G that generates L and expresses $\mathbf{L}$ and φ exists.

Complete sentences are associated to $\mathbf{Ho}$ formulas, which in most cases are $\mathbf{Fo}$ formulas, but not always: the semantics of significant fragments of languages like English requires full $\mathbf{Ho}$ [BC80]. Syntactic constituents below the sentence level are mapped to arbitrary $\mathbf{Ho}$ expressions. In addition, the definition of $\tau(\cdot)$ over grammar lexicons gives rise to their partition into:

– An arbitrarily large *content lexicon* whose (content) words, nouns ($\mathbf{N}$s like "beer"), proper nouns ($\mathbf{Pn}$s like "Max"), transitive verbs ($\mathbf{TV}$s like "drinks"), etc., stand for *individuals* and *relations*, e.g.:

$$\begin{array}{rl} \mathbf{Pn} \to \text{Max} & \tau(\mathbf{Pn}) := \lambda P^{e\to t}.P(\text{Max}) \\ \mathbf{N} \to \text{beer} & \tau(\mathbf{N}) := \lambda x^{e}.\mathit{Beer}(x) \\ \mathbf{TV} \to \text{drinks} & \tau(\mathbf{TV}) := \lambda \beta^{(e\to t)\to t}.\lambda x^{e}.\beta(\lambda y^{e}.\mathit{drinks}(x,y)) \end{array}$$

– A finite *function lexicon* whose (function) words stand for *logical operations* over individuals and relations. Relatives ($\mathbf{Relp}$s like "who") and conjunctions ($\mathbf{Crd}$s like "and") express Boolean intersection (or conjunction), determiners ($\mathbf{Det}$s like "some") express quantification, negation ("not") Boolean complementation (or negation), e.g.:

$$\begin{array}{rl} \mathbf{Det} \to \text{some} & \tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\exists x^{e}(P(x) \wedge Q(x)) \\ \mathbf{Relp} \to \text{that} & \tau(\mathbf{Relp}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\lambda x^{e}.(P(x) \wedge Q(x)) \\ \mathbf{Neg} \to \text{not} & \tau(\mathbf{Neg}) := \lambda P^{e\to t}.\lambda x^{e}\neg P(x) \\ \mathbf{Crd} \to \text{and} & \tau(\mathbf{Crd}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\lambda x^{e}.(P(x) \wedge Q(x)) \end{array}$$

Example 2.1.8. To illustrate the notions of this section we use as an example Pratt and Third's fragment of English COP defined by the following semantically enriched context free grammar:

(Phrase Structure Rules)	(Semantic actions)
$\mathbf{S} \to \mathbf{NP\ VP}$	$\tau(\mathbf{S}) := \tau(\mathbf{NP})(\tau(\mathbf{VP}))$
$\mathbf{VP} \to$ is a $\mathbf{N}$	$\tau(\mathbf{VP}) := \tau(\mathbf{N})$
$\mathbf{VP} \to$ is $\mathbf{Neg}$ a $\mathbf{N}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{N}))$
$\mathbf{NP} \to \mathbf{Pn}$	$\tau(\mathbf{NP}) := \tau(\mathbf{Pn})$
$\mathbf{NP} \to \mathbf{Det\ N}$	$\tau(\mathbf{NP}) := \tau(\mathbf{Det})(\tau(\mathbf{N}))$

(Content Lexicon)

$$\begin{array}{rl}
\mathbf{N} \rightarrow \text{woman} & \tau(\mathbf{N}) := \lambda x^{e}.\textit{Woman}(x) \\
\mathbf{N} \rightarrow \text{man} & \tau(\mathbf{N}) := \lambda x^{e}.\textit{Man}(x) \\
\mathbf{N} \rightarrow \text{person} & \tau(\mathbf{N}) := \lambda x^{e}.\textit{Person}(x) \\
\mathbf{N} \rightarrow \text{human} & \tau(\mathbf{N}) := \lambda x^{e}.\textit{Human}(x) \\
\mathbf{Pn} \rightarrow \text{Mary} & \tau(\mathbf{Pn}) := \lambda P^{e\rightarrow t}.P(\text{Mary}) \\
\vdots & \vdots
\end{array}$$

(Function Lexicon)

$$\begin{array}{ll}
\mathbf{Det} \rightarrow \text{every} & \tau(\mathbf{Det}) := \lambda P^{e\rightarrow t}.\lambda Q^{e\rightarrow t}.\forall x^{e}(P(x) \Rightarrow Q(x)) \\
\mathbf{Det} \rightarrow \text{some} & \tau(\mathbf{Det}) := \lambda P^{e\rightarrow t}.\lambda Q^{e\rightarrow t}.\exists x^{e}(P(x) \wedge Q(x)) \\
\mathbf{Det} \rightarrow \text{no} & \tau(\mathbf{Det}) := \lambda P^{e\rightarrow t}.\lambda Q^{e\rightarrow t}.\forall x^{e}(P(x) \Rightarrow \neg Q(x)) \\
\mathbf{Neg} \rightarrow \text{not} & \tau(\mathbf{Det}) := \lambda P^{e\rightarrow t}.\lambda x^{e}.\neg P(x)
\end{array}$$

As the reader can see, the lexicon of COP is divided into a finite function lexicon containing entries for "not", "some", "no" and "every", and an arbitrarily large content lexicon of common and proper nouns. To each lexical entry and each grammar rewriting rule a semantic action is associated. The words in the function lexicon express **Fo** universal and existential quantification, in addition to a (very restricted) form of negation. The content lexicon specifies a **Fo** signature composed of unary predicates and individual constants.

Suppose now we want to check whether the sentence

$$\text{Mary is a woman.} \tag{2.1}$$

is a COP sentence and whether it compositionally translates into $\textit{Woman}(\text{Mary})$.

Grammar derivations, as is typical with ordinary context-free grammars, are captured by parse trees. Figure 2.2 shows the parse tree of sentence (2.1). At each internal node, child nodes are applied to each other, in accordance with the semantic actions, and normalized. Such application(s) and normalization(s) must comply with the typing rules. For example, at the root node **S**, the following typing judgement occurs, which, modulo normalization, yields $\textit{Woman}(\text{Mary})$ as the **Fo** meaning representation of the complete utterance (note that we make the typing explicit[1]):

$$\frac{\vdash \lambda P^{e\rightarrow t}.P(\text{Mary}){:}(e{\rightarrow}t){\rightarrow}t \qquad \vdash \lambda x^{e}.\textit{Woman}(x){:}e{\rightarrow}t}{\vdash \lambda P^{e\rightarrow t}.P(\text{Mary})(\lambda x^{e}.\textit{Woman}(x)){:}t}$$

$$\lambda P^{e\rightarrow t}.P(\text{Mary})(\lambda x^{e}.\textit{Woman}(x)) \rhd \textit{Woman}(\text{Mary})$$

The computation of $\tau(\cdot)$ can be done on the fly, i.e., side-by-side with the computation of the parse tree. In a similar manner, we can say "Mary is not a man." and translate it into $\neg\textit{Man}(\text{Mary})$. In fact, we can express all these **Fo** sentences:

$\textit{Woman}(\text{Mary})$	Mary is a woman.
$\neg\textit{Man}(\text{Mary})$	Mary is not a man.
$\exists x(\textit{Person}(x) \wedge \textit{Woman}(x))$	Some person is a woman
$\exists x(\textit{Person}(x) \wedge \neg\textit{Woman}(x))$	Some person is not a woman.
$\forall x(\textit{Man}(x) \Rightarrow \textit{Person}(x))$	Every man is a person.
$\forall x(\textit{Person}(x) \Rightarrow \textit{Human}(x))$	Every person is a human.
$\forall x(\textit{Woman}(x) \Rightarrow \neg\textit{Man}(x))$	No woman is a man.

[1] In this example types are a little redundant, but we will see controlled languages later in this thesis that do exploit typing.

More in general, by suitably modifying the content lexicon, we can express *any* $\mathbf{FO}$ sentence of these forms. This defines a proper fragment of $\mathbf{FO}$, which we may denote by $\mathbf{L}_{\text{COP}}$. The signature of $\mathbf{L}_{\text{COP}}$ will be "defined" by COP's content lexicon. In the example, $\mathbf{Sig}_{\text{COP}} = (\{\textit{Mary}, \dots\}, \emptyset, \{\textit{Person}, \textit{Man}, \textit{Human}, \textit{Woman}, \dots\})$. Notice that if the content lexicon is finite *(i)* only a finite number of sentences will be generated and *(ii)* both $\mathbf{L}_{\text{COP}}$ and its signature will be finite. COP's grammar defines no recursive constituents: the number of utterances that can be generated (and the number of expressible $\mathbf{FO}$ formulas) is bounded by the size of the content lexicon [PHT06].

Notice that this greatly limits the expressiveness of COP. Formulas are built out of unary predicates, contain at most two predicate symbols, make use of at most one variable and quantifier prefix, etc. Thus, COP gives rise only to a (restricted) form of negation and conjunction. In particular, $\mathbf{L}_{\text{COP}}$ is neither closed under negation nor under conjunction and hence is not "Boolean closed". ♣

2.2 Semantic Complexity

Modulo formal semantics, the coverage of a fragment of natural language (whether controlled or not), gives rise to a certain number of computational properties, viz., to a certain number of reasoning problems associated to their meaning representations whose computational complexity can be studied. Such computational properties depend mainly on the different combinations of function words of the fragments, which stand for logical operators.

Reasoning Problems. A *decision problem* P is usually described in terms of its *input(s)* and of a *question* or property we want to verify. A *reasoning problem* is, in particular, any decision problem related to a logic $\mathbf{L}$ ($\mathbf{FO}$, $\mathbf{HO}$ or any of their fragments). Decision problems can be modelled as *languages* $\text{P} \subseteq \{0,1\}^*$, whereas algorithms can be modelled as Turing machines. A Turing machine M is said to *decide* a problem P whenever, for all $w \in \{0,1\}^*$, M halts in an accepting state if $w \in \text{P}$, and halts in a non-accepting state otherwise. Such a computation makes use of *time* and *space* resources, which can be measured as a function of the size $|w|$ of the Turing machine's input w.

Such space and time resources give rise to the *computational properties* of decision problems, viz., the so-called *decision classes* into which they can be classified. In this thesis we only deal with the basic decision classes, as studied in, e.g., [Pap94] and [GJ79]. **LSpace** is the class of problems decidable by a deterministic Turing machine using logarithmic space, **NLSpace** is the class decidable by a non-deterministic Turing machine using logarithmic space, **PTime** is the class of problems decidable by a deterministic Turing machine in polynomial time, **NPTime**[2] is the class decidable by a non-deterministic Turing machine in polynomial time, etc. [Pap94, GJ79].

The following inclusions among decision classes hold: **LSpace** $\subseteq$ **NLSpace** $\subseteq$ **PTime** $\subseteq$ **NPTime** $\subseteq$ **PSpace** $\subseteq$ **ExpTime** $\subseteq$ **NExpTime** $\subseteq$ **ExpSpace**. Furthermore, in addition to this, **coNLSpace** $=$ **NLSpace**, **coLSpace** $=$ **LSpace**, **coPTime** $=$ **PTime**, and, in general, all deterministic time decision classes and non-deterministic space decision classes above **NPTime** are closed under complement. It is commonly believed that **PTime** $\subsetneq$ **NPTime** and that **coNPTime** and **NPTime** only overlap, with none containing the other, but no proof of these conjectures has been found so far.

Given a class $\mathbf{C}$, a problem P is said to be $\mathbf{C}$-*hard* whenever for each problem $\text{P}' \in \mathbf{C}$ there exists a Turing machine M, called a *log-space reduction* (or simply *reduction*), s.t., for all $w \in \{0,1\}^*$, $w \in \text{P}$ iff $M(w) \in \text{P}'$, and that runs in space logarithmic in $|w|$. A $\mathbf{C}$-hard problem is as

[2] We use a notation different from the traditional one to avoid confusion with noun phrase constituents, $\mathbf{NP}$s.

hard as any problem in **C**, possibly harder, i.e., reductions make **C** a complexity lower bound for P. If, in addition, **C** is a complexity upper bound for P, i.e., if P can be shown to be in **C**, P is said to be **C**-complete. Since log-space reductions are closed under composition, to show that a problem is **C**-hard it suffices to reduce to P a problem P′ that is already known to be **C**-hard.

Semantic Complexity. The different combinations of function and content words covered by natural language fragments and their impact on the expressiveness of their associated (**HO** and **FO**) meaning representations give rise to the notion of semantic complexity. Such semantic complexity depends mainly on the combinations of function words covered by such fragments. In particular, fragments expressing full Boolean negation and conjunction, viz., "Boolean closed" fragments can be shown to be hard for **coNPTime** or **NPTime** for most of the reasoning problems studied in this thesis.

Definition 2.2.1 (Semantic complexity)**.** The *semantic complexity* of a fragment L of English is the class of all the computational properties of the logic reasoning problems P related to their (**FO** or **HO**) meaning representations $\tau(L)$.

Pratt and Third when introducing in [PHT06] the notion of semantic complexity for a fragment of English considered only the computational complexity of the satisfiability problem (SAT) of their meaning representations. We thus generalize their notion to cover all possible reasoning problems, which makes sense insofar as in many cases such problems do not reduce to each other.

2.3 Pratt and Third's Fragments of English

In the remainder of this thesis, we will be using as a benchmark for the semantic complexity and expressive power of controlled languages, I. Pratt and A. Third's *fragments of English*. The fragments of English aim at capTuring in English common-sense reasoning, such as, e.g., syllogistic reasoning [PHT06, PH01, PH04, Thi06], that comprises reasonings of the form

Every person is human.	$\forall x(\textit{Person}(x) \Rightarrow \textit{Human}(x))$
Some woman is a person.	$\exists x(\textit{Woman}(x) \wedge \textit{Person}(x))$
$\therefore$ Some woman is human.	$\therefore \exists x(\textit{Woman}(x) \wedge \textit{Human}(x))$

The fragments of English are built "incrementally" using semantically enriched context-free grammars. A base fragment, COP, devoid of recursive constituents and covering **N**s, **Pn**s, **VP** negation, and the **Det**s "some", "every" and "no" is first defined (recall Example 2.1.8). Each successive fragments of English extends COP's coverage to a fresh English construct (**TV**s, **Relp**s, anaphora, etc.). Table 2.2 summarizes their coverage of English. Notice that negation expresses full Boolean negation. Relative clauses introduce both conjunction and recursion. Thus, each fragment above COP+Rel is "Boolean closed". Finally, each fragment gives rise to a fragment of **FO**, as outlined in Table 2.1. For a detailed definition we send the reader to [PHT06].

Consider now the *satisfiability* reasoning problem (SAT) defined by

- **Input:** a formula φ from **L**.
- **Question:** does a model for φ exists?

The SAT problem for the COP fragment can be shown to be in **NLSpace**. This is because of its limited expressiveness. By contrast, "Boolean closed" fragments of English (COP+Rel can express the propositional calculus) are already **NPTime**-hard and even harder (**ExpTime**-hard) when, in addition, **TV**s are considered. Unrestricted anaphoric pronouns make SAT undecidable (by a reduction from the undecidable unbounded tiling problem [PHT06]). Table 2.2 summarizes the semantic complexity for SAT of the fragments of English.

COP	$\psi_l(x) \rightarrow P(x)$	$\forall x(\psi_l(x) \Rightarrow \pm\psi_r(x))$	No student failed.
	$\psi_r(x) \rightarrow \pm\psi_l(x)$	$\exists x(\psi_l(x) \wedge \psi_r(x))$	A student failed.
COP+	$\psi_l(x) \rightarrow P(x)$	$\forall x(\psi_l(x) \Rightarrow \pm\psi_r(x))$	No student failed.
TV	$\psi_r(x) \rightarrow \pm\psi_l(x) \mid \forall y(P(x) \Rightarrow \pm\Psi(x,y))$	$\exists x(\psi_l(x) \wedge \psi_r(x))$	Some student follows
	$\mid \exists y(P(x) \wedge \Psi(x,y))$		every course.
COP+	$\psi_l(x) \rightarrow P(x)$	$\forall x(\psi_l(x) \Rightarrow \pm\psi_r(x))$	Every student
TV+	$\psi_{tv}(x) \rightarrow \pm\psi_l(x) \mid \forall y(P(x) \Rightarrow \pm\Psi(x,y))$		gives no credit
DTV	$\mid \exists y(P(x) \wedge \psi(x,y))$		to some student.
	$\psi_{dtv}(x,y) \rightarrow \forall z(A(x) \Rightarrow \pm\Xi(x,y,z))$	$\exists x(\psi_l(x) \wedge \psi_r(x))$	A student
	$\mid \exists z(P(x) \wedge \Xi(x,y,z))$		borrowed a book
	$\psi_r(x) \rightarrow \psi_{tv}(x) \mid \forall y(P(x) \Rightarrow \pm\psi_{dtv}(x,y))$		from some library.
	$\mid \exists y(P(x) \wedge \psi_{dtv}(x,y))$		
COP+	$\psi_l(x) \rightarrow P(x) \mid \pm\psi_l(x) \wedge \pm\psi_l(x)$	$\forall x(\pm\psi_l(x) \Rightarrow \pm\psi_r(x))$	Every student who is not
Rel	$\psi_r(x) \rightarrow \psi_l(x)$	$\exists x(\pm\psi_l(x) \wedge \pm\psi_r(x))$	dumb is smart.
COP+	$\psi_l(x) \rightarrow P(x) \mid \pm\psi_r(x) \wedge \pm\psi_r(x)$	$\forall x(\psi_l(x) \Rightarrow \pm\psi_r(x))$	No student failed.
TV+	$\psi_r(x) \rightarrow \pm\psi_l(x) \mid \forall y(P(x) \Rightarrow \pm\Psi(x,y))$	$\exists x(\psi_l(x) \wedge \psi_r(x))$	Some student studies
Rel	$\mid \exists y(P(x) \wedge \Psi(x,y))$		every course.
COP+	$\psi_l(x) \rightarrow A(x) \mid \pm\psi_r \wedge \pm\psi_r$	$\forall x(\psi_l(x) \Rightarrow \pm\psi_r(x))$	Every helpful student
TV+	$\psi_{tv}(x) \rightarrow \pm\psi_l(x) \mid \forall y(A(x) \Rightarrow \pm\Psi(x,y))$		gives some aid
DTV+	$\mid \exists y(A(x) \wedge \Psi(x,y))$		to some student.
Rel	$\psi_{dtv}(x,y) \rightarrow \forall z(A(x) \Rightarrow \pm\Xi(x,y,z))$	$\exists x(\psi_l(x) \wedge \psi_r(x))$	Some diligent student
	$\mid \exists z(A(x) \wedge \Xi(x,y,z))$		borrowed every book
	$\psi_r(x) \rightarrow \psi_{tv}(x) \mid \forall y(A(x) \Rightarrow \pm\psi_{dtv}(x,y))$		from every library.
	$\mid \exists y(A(x) \wedge \psi_{dtv}(x,y))$		

Table 2.1: The meaning representations generated by the fragments of English. Note that $\Psi(x,y)$ (resp. $\Xi(x,y,z)$) stands for some binary (resp. ternary) atom, while $\pm$ means that a formula may or may not be negated. Complete utterances comply with the pattern $\mathbf{Det\ N\ VP}$, where $\mathbf{Det}$ maps, modulo $\tau(\cdot)$, into either $\forall$ or $\exists$, $\mathbf{N}$ into $\psi_l(x)$, the subject, and $\mathbf{VP}$ into $\psi_r(x)$, the predicate [PHT06]. Relatives introduce recursion and conjunction.

COP	Copula, common and proper nouns, negation, universal and existential quantifiers
COP+Rel	COP plus relative pronouns
COP+TV	COP plus transitive verbs
COP+TV+ DTV	COP+TV plus ditransitive verbs
COP+TV+ DTV+RA	COP+TV+DTV plus anaphoric pronouns (e.g., he, him, it, herself)

COP	in **NLSpace**
COP+TV	**NLSpace**-complete
COP+DTV	in **PTime**
COP+TV+DTV	in **PTime**
COP+Rel	**NPTime**-complete
COP+TV+Rel	**ExpTime**-complete
COP+TV+Rel+RA	**ExpTime**-complete
COP+DTV+Rel	**NExpTime**-complete
COP+TV+DTV+Rel	**NExpTime**-complete
COP+TV+Rel+GA	undecidable
COP+TV+DTV+Rel+RA	undecidable
COP+TV+DTV+Rel+GA	undecidable

Table 2.2: Above: Coverage of the main fragments of English (the other fragments are obtained by combining them together). **Below:** Their semantic complexity w.r.t. SAT [PHT06, PH09, PHM09].

2.4 Summary

In this chapter we have briefly recalled the theory of formal and computational semantics for natural languages and their fragments, and the ensuing notions of semantically enriched grammars, meaning representations and semantic complexity. Since natural language meaning representations are based on $\mathbf{Ho}$ and $\mathbf{Fo}$, we defined the syntax and the semantics of both logics. The notation introduced there will be used throughout the remainder of this thesis. We also showed how formal semantics enforces clustering natural language words into content words, expressing relations and individuals, and function words, expressing operations over such individuals and relations. In particular, the different coverage of function words by fragments (controlled or not) of natural languages like English has a crucial impact on their semantic complexity. This is because, modulo compositional translations, those fragments express or map to logical fragments (of $\mathbf{Fo}$ and $\mathbf{Ho}$), and their function words map to logical operators or constructors.

Chapter 3

Ontology Languages and Conjunctive Queries

In this chapter we give an overview of description logic ontologies and knowledge bases, which formally underpin OWL ontologies and OBDASs. We also provide some background on formal query languages and on data complexity. Last, but least, we explain how ontology and formal query languages can be used to understand the scalability of controlled language interfaces by expressing them in controlled English. This means: defining semantically enriched grammars (and hence controlled fragments of English), equipped with a compositional translation $\tau(\cdot)$, that express those ontology and formal query languages. With this technique we can exploit the computational properties of the ontology and query languages to study the semantic complexity of controlled languages and their scalability to data. This chapter is derived mainly from [BCN+03], Chapters 2 and 3, [AHV95], Chapters 4 and 17, and [CGL+07].

3.1 Description Logic Ontologies

In an OBDAS, an ontology provides a conceptual view on the data stored in a database, which can be accessed by formulating formal queries over the ontology. Ontologies are formally underpinned by description logics, which are a family of knowledge representation formalisms based on decidable fragments of **FO** [BCN+03]. In description logics, the domain of interest is structured in terms of instances (standing for the individuals in the domain), concepts (standing for classes of individuals/instances) and roles (standing for binary relations among individuals/instances). The instances, concepts and roles provide the basic vocabulary of the domain. Concepts and roles are then combined together into sets of constraints that hold among domain instances (i.e., a logical theory) which give rise to concept and role hierarchies. Description logics are also known as ontology languages. Formal queries are, typically **FO** (or **HO**) formulas that specify an information request. The semantics of query evaluation in OBDASs is based on **FO** entailment: the system checks if the query is logically entailed by the ontology and the database.

3.1.1 Ontology Languages and *DL-Lite*

The OWL-DL fragment of OWL is underpinned by the description logic $\mathcal{SROIQ}$. We are interested in description logics of different expressiveness, ranging from the *DL-Lite* family [CGL+07] to $\mathcal{ALCHQI}$. The *DL-Lite* family [CdL+06, CdV+06, CdL+05a] is a family of ontology languages optimized for data access in OBDASs. They can be defined as fragments of $\mathcal{ALCHQI}$, by suitably restricting its syntax.

Syntax	Semantics
c	$c^{\mathcal{I}} \in \mathbb{D}_{\mathcal{I}}$
A	$A^{\mathcal{I}} \subseteq \mathbb{D}_{\mathcal{I}}$
$\exists_{\geq k} R{:}C$	$(\exists_{\geq k} R{:}C)^{\mathcal{I}} := \{c \mid \#(\{c' \text{ s.t. } (c,c') \in R^{\mathcal{I}} \text{ and } c' \in C^{\mathcal{I}}\}) \geq k\}$
$\neg C$	$(\neg C)^{\mathcal{I}} := \mathbb{D}_{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap C'$	$(C \sqcap C')^{\mathcal{I}} := C^{\mathcal{I}} \cap C'^{\mathcal{I}}$
r	$r^{\mathcal{I}} \subseteq \mathbb{D}_{\mathcal{I}} \times \mathbb{D}_{\mathcal{I}}$
r^-	$(r^-)^{\mathcal{I}} := \{(d,d') \mid (d',d) \in r^{\mathcal{I}}\}$
$C \sqsubseteq C'$	$\mathcal{I} \models C \sqsubseteq C'$ iff $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$
$R \sqsubseteq R'$	$\mathcal{I} \models R \sqsubseteq R'$ iff $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$
$A(c)$	$\mathcal{I} \models A(c)$ iff $c^{\mathcal{I}} \in A^{\mathcal{I}}$
$r(c,c')$	$\mathcal{I} \models r(c,c')$ iff $(c^{\mathcal{I}}, c'^{\mathcal{I}}) \in r^{\mathcal{I}}$
$\mathcal{O}$	$\mathcal{I} \models \mathcal{O}$ iff for all $\alpha \in \mathcal{O}, \mathcal{I} \models \alpha$
$\mathcal{D}$	$\mathcal{I} \models \mathcal{D}$ iff for all $\alpha \in \mathcal{D}, \mathcal{I} \models \alpha$
$(\mathcal{O},\mathcal{D})$	$\mathcal{I} \models (\mathcal{O},\mathcal{D})$ iff $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \mathcal{D}$

Table 3.1: Semantics of $\mathcal{ALCHQI}$ (and its fragments).

The description logic $\mathcal{ALCHQI}$. In the $\mathcal{ALCHQI}$ description logic, *concepts* C and *roles* R are formed according to the following syntax:

$$\begin{array}{rcl} C & \rightarrow & A \mid \exists_{\leq k} R{:}C \mid \neg C \mid C \sqcap C \\ R & \rightarrow & r \mid r^- \end{array}$$

where A stands for an atomic concept or *concept name* (a unary predicate), r for a *role name* (a binary predicate) and r^- for its inverse. We can enrich the set of $\mathcal{ALCHQI}$ concepts, modulo the following (explicit) definitions:

$$\begin{array}{rclcrcl} \bot & := & \neg A \sqcap A & & C \sqcup C' & := & \neg(\neg C \sqcap \neg C') \\ \exists_{\geq k} R{:}C & := & \neg(\exists_{\leq k+1} R{:}\neg C) & & \exists R & := & \exists R{:}\top \\ \exists R{:}C & := & \exists_{\geq 1} R{:}C & & \forall R{:}C & := & \neg(\exists R{:}\neg C) \\ \exists_{=k} & := & \exists_{\leq k} R{:}C \sqcap \exists_{\geq k} R{:}C & & \top & := & \neg\bot \end{array}$$

where A is some atomic concept.

In a $\mathcal{ALCHQI}$ ontology $\mathcal{O}$, intensional knowledge is specified by means of a set of *assertions* α, viz.,

- *concept inclusions* of the form $C \sqsubseteq C'$, stating IS-A (set inclusion) between the instances of the concepts C and C', and
- *role inclusions* of the form $R \sqsubseteq R'$, stating IS-A (set inclusion) among role instances.

The *size* $\#(\mathcal{O})$ of an ontology $\mathcal{O}$ where $\#(S)$ denotes the cardinality of set S, is given by the number of assertions it contains.

A *database*, expressing extensional knowledge, is a finite set $\mathcal{D}$ of unary and binary ground atoms of the form $A(c)$, $r(c,c')$, where c, c' are individual constants. The *active domain* $adom(\mathcal{D})$ of a database $\mathcal{D}$ is the set of all the pairwise distinct constants that occur among the ground atoms of $\mathcal{D}$. The *size* of a database $\mathcal{D}$ is given by $\#(adom(\mathcal{D}))$.

A *knowledge base* is a pair $(\mathcal{O},\mathcal{D})$, where $\mathcal{O}$ is an ontology and $\mathcal{D}$ a database.

We consider frames $\mathbf{Dom}$ constituted of countably many individual *object names* or individual constants, i.e., $\mathbf{Dom} \subseteq \mathbf{C}$, where $\mathbf{C}$ is countably infinite. This semantic assumption for ontology languages is known the literature as the *standard domain assumption* (SDA).

Given $\mathbf{Dom}$, the semantics of concepts, assertions, ontologies and knowledge bases is specified by considering $\mathbf{FO}$ interpretations $\mathcal{I}$ where $\mathbb{D}_\mathcal{I} \subseteq \mathbf{Dom}$ and the interpretation function $\cdot^\mathcal{I}$ maps *(i)* concept names A and roles r into, resp., subsets of the domain and of its cross product, and *(ii)* object names c to elements of the domain. It can be extended to complex concepts C and roles R by structural recursion as shown in the first part of Table 3.1.

An interpretation $\mathcal{I}$ is said to be a *model* of a concept inclusion $C \sqsubseteq C'$, role inclusion $\mathcal{I} \models R \sqsubseteq R'$ or membership $A(c)$, $r(c, c')$ assertion when, resp., $\mathcal{I} \models C \sqsubseteq C'$, $\mathcal{I} \models R \sqsubseteq R'$, or $\mathcal{I} \models A(c)$ and $\mathcal{I} \models r(c, c')$. It is said to be a *model* of an ontology $\mathcal{O}$ or a database $\mathcal{D}$ when, resp. $\mathcal{I} \models \mathcal{O}$ or $\mathcal{I} \models \mathcal{D}$. It is said, finally, to be a *model* of a knowledge base $(\mathcal{O}, \mathcal{D})$ when $\mathcal{I} \models (\mathcal{O}, \mathcal{D})$, i.e., when it is a model of $\mathcal{O}$ and $\mathcal{D}$. See the second and third parts of Table 3.1.

Two concepts C and C' are said to be *equivalent* iff, for all $\mathcal{I}$, $C^\mathcal{I} = C'^\mathcal{I}$. An assertion α is to *imply* α', in symbols $\alpha \models \alpha'$, iff, for all interpretations $\mathcal{I}$, $\mathcal{I} \models \alpha$ implies $\mathcal{I} \models \alpha'$. When the converse also holds, we say that they are *equivalent*, in symbols $\alpha \equiv \alpha'$. These notions can be generalized to ontologies in the obvious way.

The semantics of $\mathcal{ALCHQI}$ allows us to introduce *global functionality* assertions of the form (*funct* R) stating that any instance or object falling under R's domain is connected to *at most one* R-successor, since

$$(\textit{funct } R) \;\equiv\; \top \sqsubseteq \exists_{\leq 1} R{:}\top$$

is a trivial consequence of $\mathcal{ALCHQI}$ semantics.

The Fragments $\mathcal{ALCI}$ and $\mathcal{ELI}$. The description logic $\mathcal{ALCI}$ is the fragment of $\mathcal{ALCHQI}$ with syntax

$$\begin{array}{rcl} C & \rightarrow & A \mid \exists R{:}C \mid C \sqcap C \mid \neg C \\ R & \rightarrow & r \mid r^- \end{array}$$
$$A(c), r(c, c')$$
$$C \sqsubseteq C'$$

Notice that $\mathcal{ALCI}$ is closed under negation. In other words, in $\mathcal{ALCI}$ we disallow concepts of the form $\exists_{\leq k} R{:}C$ (called *qualified number restrictions*) but introduce explicitly $\exists R{:}C$ (called *qualified existential*). Moreover, we disallow role inclusions in the ontology.

By disallowing further concept negation we can define the description logic $\mathcal{ELI}$, with syntax

$$\begin{array}{rcl} C & \rightarrow & A \mid \exists R{:}C \mid C \sqcap C \\ R & \rightarrow & r \mid r^- \end{array}$$
$$A(c), r(c, c')$$
$$C \sqsubseteq C'$$

which we can extend with *universal restrictions* to the right of $\sqsubseteq$ since

$$A \sqsubseteq \forall r{:}A' \;\equiv\; \exists r^-{:}A \sqsubseteq A'.$$

The *DL-Lite* family. In the *DL-Lite* family of description logics we restrict the syntax and constructors of the concepts C_r to the *right* and C_l to the *left* of the *subsumption* symbol $\sqsubseteq$. In addition, ontology assertions are also restricted. Different restrictions give way to different description logics. It is precisely these restrictions which give rise to the good computational properties for data access that they exhibit, as we shall see later in this chapter..

Table 3.2 shows the four basic members of the *DL-Lite* family, viz., *DL-Lite*$_{core}$ (the core fragment), *DL-Lite*$_R$, *DL-Lite*$_F$ and *DL-Lite*$_\sqcap$. The essential features of *DL-Lite*$_{core}$ are: *(i)* only unqualified existential roles $\exists R$ are admitted *(ii)* negation is restricted to *right* concepts, without

DL-Lite$_{core}$	*DL-Lite*$_R$
$C_l \rightarrow A \mid \exists R$ $C_r \rightarrow \neg A \mid \neg\exists R \mid C_l \mid C_r \sqcap C'_r$ $R \rightarrow r \mid r^-$ $A(c), r(c, c')$ $C_l \sqsubseteq C_r$	$C_l \rightarrow A \mid \exists R$ $C_r \rightarrow \neg A \mid \neg\exists R \mid C_l \mid C_r \sqcap C'_r$ $R \rightarrow r \mid r^-$ $A(c), r(c, c')$ $C_l \sqsubseteq C_r$ $R \sqsubseteq R'$
DL-Lite$_F$	*DL-Lite*$_\sqcap$
$C_l \rightarrow A \mid \exists R$ $C_r \rightarrow \neg A \mid \neg\exists R \mid C_l \mid C_r \sqcap C'_r$ $R \rightarrow r \mid r^-$ $A(c), r(c, c')$ $C_l \sqsubseteq C_r$ (*funct* R)	$C_l \rightarrow A \mid \exists R \mid C_l \sqcap C'_l$ $C_r \rightarrow \neg A \mid \neg\exists R \mid C_l \mid C_r \sqcap C'_r$ $R \rightarrow r \mid r^-$ $A(c), r(c, c')$ $C_l \sqsubseteq C_r$

Table 3.2: The *DL-Lite* family.

closing them (it can only be applied to atomic or unqualified existential roles), *(iii)* conjunction is restricted to right concepts, *(iv)* no number restrictions are allowed, and *(v)* no role inclusions or role functionality assertions are allowed.

The other three description logics extend *DL-Lite*$_{core}$ by, resp., adding role assertions, functionality assertions and by closing left concepts under conjunction. The remaining members of the family, *DL-Lite*$_{R,\sqcap}$, *DL-Lite*$_{F,R}$ and *DL-Lite*$_{F,\sqcap}$ are built by pairwise merging the basic fragments.

Figure 3.1 below shows the resulting lattice of inclusions. We have highlighted (in black) the sublattice that we will study in the following chapters. As the figure shows, they are all subsumed by $\mathcal{ALCHQI}$, but are, in general, incomparable with $\mathcal{ELI}$ and/or $\mathcal{ALCI}$ (although both *DL-Lite*$_{core}$ and *DL-Lite*$_\sqcap$ can be subsumed by $\mathcal{ALCI}$).

We can extend the syntax of *DL-Lite*$_{core}$ with: *(i)* conjunction $C_r \sqcap C'_r$ among right concepts, *(ii)* disjunction $C_l \sqcup C'_l$ among left concepts, since

$$C_l \sqsubseteq C_r \sqcap C'_r \equiv \{C_l \sqsubseteq C_r, C_l \sqsubseteq C'_r\},$$
$$C_l \sqcup C'_l \sqsubseteq C_r \equiv \{C_l \sqsubseteq C_r, C'_l \sqsubseteq C_r\},$$

and the *falsum (iii)* $\bot$ among right concepts, since

$$C_l \sqsubseteq \bot \equiv C_l \sqsubseteq C_r \sqcap \neg C_r,$$

where C_r is a "fresh" concept. Moreover, we can extend the syntax of *DL-Lite*$_R$ with right *qualified existential roles*, since

$$C_l \sqsubseteq \exists R{:}C_r \equiv \{C_l \sqsubseteq \exists R, R \sqsubseteq R', \exists R'^- \sqsubseteq C_r\},$$

where R' is a "fresh" role.

Note that for the *DL-Lite* family, two additional semantic assumptions can hold. Given a database $\mathcal{D}$, every interpretation $\mathcal{I}$ must map any pair of distinct constants $c, c' \in$ *adom*$(\mathcal{D})$ to distinct elements of its domain $\mathbb{D}_\mathcal{I}$. This assumption is known as the *unique name assumption* (UNA). In addition, we may also enforce that for all $c \in$ *adom*$(\mathcal{D})$, $c^\mathcal{I} := c$. This stronger assumption is known as the *standard names assumption* (SNA).

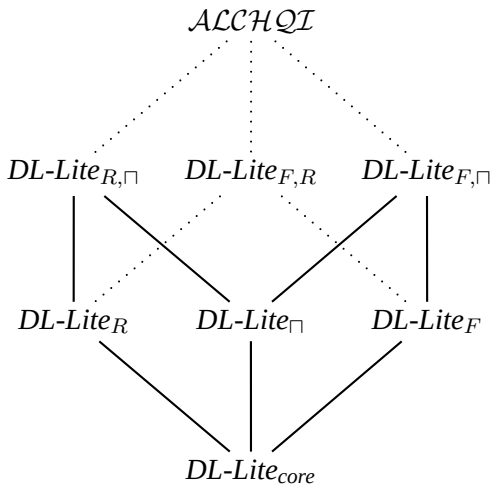


Figure 3.1: The relative expressive power of the *DL-Lite* family.

3.1.2 Basic Properties

The description logics presented in this chapter are contained in the 2-variable fragment of $\mathbf{FO}$ with counting quantifiers, $\mathbf{C}^2$. This can be made explicit by means of the $\cdot^{t_{x,y}}$ and $\cdot^{t_{y,x}}$ translations. We define the translations $\cdot^{t_{x,y}}$ and $\cdot^{t_{y,x}}$ from $\mathcal{ALCHQI}$ to $\mathbf{C}^2$ by mutual recursion on concepts and roles as follows:

$$R^{t_{x,y}} := \begin{cases} r(x,y), \text{ if } R = r \\ R^{t_{y,x}}, \text{ otherwise} \end{cases} \qquad R^{t_{y,x}} := \begin{cases} r(y,x), \text{ if } R = r^- \\ R^{t_{x,y}}, \text{ otherwise} \end{cases}$$

$$\begin{aligned} A^{t_{x,y}} &:= A(x) & A^{t_{y,x}} &:= A(y) \\ (\neg C)^{t_{x,y}} &:= \neg C^{t_{x,y}} & (\neg C)^{t_{y,x}} &:= \neg C^{t_{y,x}} \\ (C \sqcap C')^{t_{x,y}} &:= C^{t_{x,y}} \wedge C'^{t_{x,y}} & (C \sqcap C')^{t_{y,x}} &:= C^{t_{y,x}} \wedge C'^{t_{y,x}} \\ (\exists_{\leq k} R{:}C)^{t_{x,y}} &:= \exists_{\leq k} y(R^{t_{x,y}} \wedge C^{t_{y,x}}) & (\exists_{\geq k} R{:}C)^{t_{y,x}} &:= \exists_{\geq k} x(R^{t_{x,y}} \wedge C^{t_{x,y}}) \end{aligned}$$

We can extend $\cdot^{t_{x,y}}$ (resp. $\cdot^{t_{x,y}}$) to ontology assertions with:

$$\begin{aligned} (C \sqsubseteq C')^{t_{x,y}} &:= \forall x(C^{t_{x,y}} \Rightarrow C'^{t_{x,y}}) \\ (R \sqsubseteq R')^{t_{x,y}} &:= \forall x \forall y(R^{t_{x,y}} \Rightarrow R'^{t_{x,y}}) \end{aligned}$$

Finally, we extend $\cdot^{t_{x,y}}$ (resp. $\cdot^{t_{y,x}}$) to ontologies as follows: $\mathcal{O}^{t_{x,y}} := \{\alpha^{t_{x,y}} \mid \alpha \in \mathcal{O}\}$. Since databases are sets $\mathbf{FO}$ atomic sentences, there is nothing to do in that case.

Notice that all such $\mathbf{C}^2$ formulas are guarded and belong thus to the 2-variable guarded fragment of $\mathbf{FO}$ with counting quantifiers, $\mathbf{GC}^2$ [BCN$^+$03]. For description logics such as $\mathcal{ALCI}$, $\mathcal{ELI}$ and the *DL-Lite* description logics without functionality assertions, the translation rules involving qualified number restrictions can be replaced by the following ones involving only qualified existentials

$$(\exists R{:}C)^{t_{x,y}} := \exists y(R^{t_{x,y}} \wedge C^{t_{y,x}}) \qquad (\exists R{:}C)^{t_{y,x}} := \exists x(R^{t_{y,x}} \wedge C^{t_{x,y}})$$

which do not require counting quantifiers. As a result, these logics are contained in the two-variable guarded fragment of $\mathbf{FO}$, $\mathbf{GF}^2$.

Proposition 3.1.1 ([BCN$^+$03])**.** *An $\mathcal{ALCHQI}$ concept C is satisfiable iff $C^{t_{x,y}}$ (resp. $C^{t_{y,x}}$) is satisfiable.*

Proof. It can be easily shown by induction on C that, for some interpretation $\mathcal{I}$, $c \in \mathbb{D}_{\mathcal{I}}$ and assignment γ:

$$c \in C^{\mathcal{I}} \text{ iff } \mathcal{I}, \gamma[x := c] \models C^{t_{x,y}},$$

and, resp., for $\cdot^{t_{x,y}}$. This closes the proof. □

The *negation normal form* $\cdot^{NNF}$ transformation of an $\mathcal{ALCI}$ concept C is defined by structural recursion on C with

$$\begin{array}{rlrl} A^{NNF} &:= A & (\neg A)^{NNF} &:= \neg A \\ (\exists R{:}C)^{NNF} &:= \exists R{:}C^{NNF} & (\neg(\exists R{:}C))^{NNF} &:= \forall R{:}(\neg C)^{NNF} \\ (C \sqcap C')^{NNF} &:= C^{NNF} \sqcap C^{NNF} & (\neg(C \sqcap C'))^{NNF} &:= (\neg C)^{NNF} \sqcup (\neg C')^{NNF} \end{array}$$

and

$$(\neg(\neg C))^{NNF} \quad := \quad C^{NNF}$$

An $\mathcal{ALCHQI}$ concept C is said *to be in negation normal form* (NNF) whenever $C^{NNF} = C$. It immediately follows:

Proposition 3.1.2 ([BCN+03])**.** *For every concept C in $\mathcal{ALCI}$, there exists an equivalent $\mathcal{ALCI}$ concept C' in NNF, and if C is in NNF, then negation is restricted to atomic concepts.*

3.2 Conjunctive and Tree-Shaped Queries

We use queries to retrieve information from OBDASs. As query languages, we consider *conjunctive queries* (CQs) and *tree shaped conjunctive queries* (TCQs) which are those CQs built using only unary and binary relations and that are tree-isomorphic. By default, and unless explicitly stated otherwise, formal queries are built over the signature $\mathbf{Sig} := (\emptyset, \{A_i \mid i \in \mathbb{N}\}, \{r_j \mid j \in \mathbb{N}\})$ of ontology language concept names and role names.

Definition 3.2.1 (Conjunctive queries and their unions)**.** A *conjunctive query* is an existentially quantified conjunction of positive $\mathbf{FO}$ relational atoms

$$\varphi(\bar{x}) := \exists \bar{y} \varphi(\bar{x}, \bar{y}) \tag{CQ}$$

over variables $\bar{x}$ and $\bar{y}$, where the free variables $\bar{x}$ are known also as the CQ's *distinguished variables*. The *length* $|\bar{x}|$ of the sequence of distinguished variables is known as the *arity* of the CQ. A *union of conjunctive queries* is a disjunction

$$\varphi(\bar{x}) := \exists \bar{y}_1 \varphi_1(\bar{x}, \bar{y}_1) \vee \cdots \vee \exists \bar{y}_k \varphi_1(\bar{x}, \bar{y}_k) \tag{UCQ}$$

of CQs, all of the same arity. CQs and UCQs are said to be *boolean* when they contain no free variables. The integer $|\varphi|$ denotes the *size* of UCQ φ, i.e., its number of symbols.

Intuitively, non-distinguished variables combined with the relational conjunctions stand for relational database table joins and selections, and distinguished variables for the information we want to project in the result: UCQs thus constitute a declarative specification of a SQL `SELECT-PROJECT-JOIN-UNION` query result table (see [AHV95], Chapter 7). We will write φ instead of $\varphi(\bar{x})$ whenever the free variables are clear from context.

Given a database $\mathcal{D}$, $\mathcal{I}(\mathcal{D})$ denotes the *interpretation associated with* $\mathcal{D}$, viz., the (Herbrand) interpretation obtained by interpreting each relation symbol S of arity k occurring in $\mathcal{D}$ (hence, in particular, concept names A and role names r) by $S^{\mathcal{I}(\mathcal{D})} := \{(c_1, \ldots, c_k) \in \mathit{adom}(\mathcal{D})^n \mid S(c_1, \ldots, c_k) \in \mathcal{D}\}$.

Let φ be a UCQ and $\mathcal{D}$ a database. Answers to databases are based on the $\mathbf{FO}$ assignments $\gamma\colon \mathit{FV}(\varphi) \to \mathbb{D}_{\mathcal{I}(\mathcal{D})}$ which are satisfying over $\mathcal{I}(\mathcal{D})$. An assignment γ is said to *satisfy* φ w.r.t. $\mathcal{D}$ whenever $\mathcal{I}(\mathcal{D}), \gamma \models \varphi$. We denote by $\mathit{Sat}_{\mathcal{D}}(\varphi)$ the set of *satisfying assignments* for φ over $\mathcal{I}(\mathcal{D})$.

Definition 3.2.2 (Database answers)**.** The set of *answers* to a UCQ φ of distinguished variables $\bar{x}$ over a database $\mathcal{D}$ is defined as

$$\mathit{ans}(\varphi, \mathcal{D}) := \{\gamma(\bar{x}) \in \mathbb{D}^{|\bar{x}|}_{\mathcal{I}(\mathcal{D})} \mid \text{there exists } \gamma \in \mathit{Sat}_{\mathcal{D}}(\varphi)\}. \qquad \textbf{(Ans)}$$

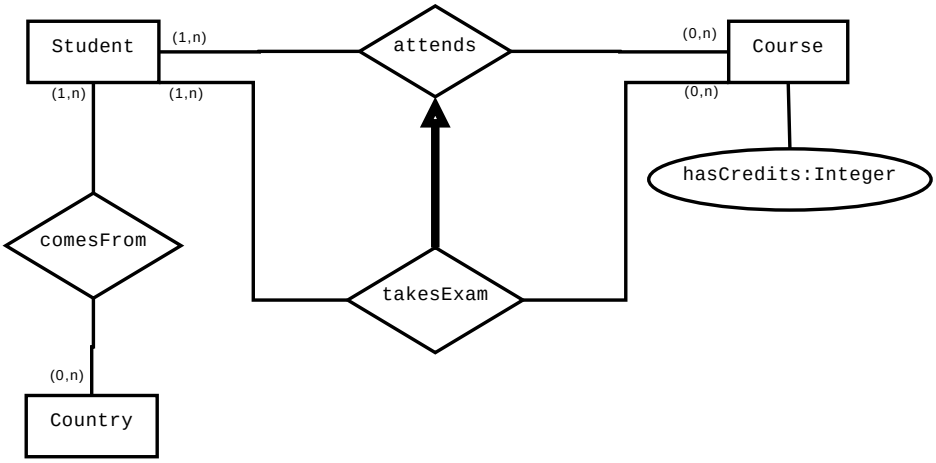
Let φ be a UCQ, and $(\mathcal{O}, \mathcal{D})$ a knowledge base. Answers to knowledge bases are based on the $\mathbf{Fo}$ groundings $\sigma\colon \mathit{FV}(\varphi) \to \mathbf{Dom}$ that guarantee φ's being logically entailed by $(\mathcal{O}, \mathcal{D})$, i.e., s.t. $(\mathcal{O}, \mathcal{D}) \models \varphi\sigma$. We say in that case that such groundings are *certain*. We denote by $\mathit{Sat}^{\mathcal{O}}_{\mathcal{D}}(\varphi)$ the set of certain groundings for φ over $(\mathcal{O}, \mathcal{D})$.

Definition 3.2.3 (Certain answers)**.** The set of *certain answers* to a UCQ φ of distinguished variables $\bar{x}$ over a knowledge base $(\mathcal{O}, \mathcal{D})$ is defined as

$$\mathit{cert}(\varphi, \mathcal{O}, \mathcal{D}) := \{\sigma(\bar{x}) \in \mathit{adom}(\mathcal{D})^{|\bar{x}|} \mid \text{there exists } \sigma \in \mathit{Sat}^{\mathcal{O}}_{\mathcal{D}}(\varphi)\}. \qquad \textbf{(Cert)}$$

Remark that this implies that, when we ask a *boolean* (U)CQ φ to a database $\mathcal{D}$ or a knowledge base $(\mathcal{O}, \mathcal{D})$ we will get as answer or, resp., as certain answer, $\{()\}$, viz., the *empty tuple* whenever $\mathcal{I}(\mathcal{D}) \models \varphi$ or, resp., $(\mathcal{O}, \mathcal{D}) \models \varphi$. Otherwise, we will get an empty set of answers or certain answers.

Example 3.2.4. Answering a UCQ over an OBDAS, i.e., returning a certain answer, exploits logical reasoning to "complete" the missing and/or incomplete factual information it may contain. Consider the following ER diagram for the student domain:



It states, basically, IS-A among two relation types, *takesExam* and *attends*, which hold among the entity types *Student* and *Course*, and to which cardinality constraints have been ascribed[1].

The *DL-Lite* family of description logics can capture, notwithstanding its simplicity, the main features of conceptual modelling languages, such as ER diagrams (and which constitute an alternative standard notation for ontologies). It is captured by the *DL-Lite*$_{R,\sqcap}$ ontology $\mathcal{O}_s$. Entities correspond to concepts, relations to roles, and IS-A and cardinality constraints to assertions:

$$\mathit{Student} \sqsubseteq \exists \mathit{attends} \quad \mathit{takesCourse} \sqsubseteq \mathit{attends}$$
$$\exists \mathit{attends} \sqsubseteq \mathit{Course} \quad \exists \mathit{attends}^{-} \sqsubseteq \mathit{Student}$$

Let now $\mathcal{D}_s$ be the database

$$\mathit{takesCourse}(\mathrm{Jay}, \mathrm{TOC}) \quad \mathit{Student}(\mathrm{Joe})$$

[1]The pair of integers (n, m) to the left (resp., right) of a relation type state that the cardinality of its domain (resp., range) ranges between n and m.

Suppose we want now to ask to the knowledge base/OBDAS $(\mathcal{O}_s, \mathcal{D}_s)$ whether Joe studies something and whether Jay is a student, viz.,

$$\varphi_s := \exists y\, \textit{attends}(\text{Joe}, y) \quad \text{and} \quad \varphi'_s := \exists y\, \textit{Student}(\text{Jay})$$

Clearly, a lot of explicit information is missing from $\mathcal{D}_s$. However, in both cases, the certain answers semantics returns a positive answer. Indeed, $(\mathcal{O}_s, \mathcal{D}_s) \models \varphi_s$ and $(\mathcal{O}_s, \mathcal{D}_s) \models \varphi'_s$. Hence, $\textit{cert}(\mathcal{O}_s, \mathcal{D}_s, \varphi_s) = \textit{cert}(\mathcal{O}_s, \mathcal{D}_s, \varphi'_s) = \{()\}$.

Observe that, in contrast to DBMSs, queries are formulated over the roles and concepts of *both* the ontology and the database. Moreover, databases in OBDASs provide only an incomplete specification of the explicit information known about the domain, which is to be completed via the implicit knowledge contained in the ontology through logical reasoning.

In the example, the *attends* role is empty in $\mathcal{D}_s$. However, since any subsequent "state" (or model) $\mathcal{D}' \supseteq \mathcal{D}_s$ must comply with $\mathcal{O}_s$ (i.e., $\mathcal{I}(\mathcal{D}') \models \mathcal{O}_s$), and in particular with the role inclusion assertion *takesCourse* $\sqsubseteq$ *attends*, the tuples in role *takesCourse* can be "propagated" (via $\mathbf{FO}$ entailment and certain answers) to the role *attends*, thus allowing the $(\mathcal{O}_s, \mathcal{D}_s)$ system to answer the query φ_s. ♣

A well-known property of the certain answers of an UCQ φ over a knowledge base $(\mathcal{O}, \mathcal{D})$, is that it can be characterized in terms of the answers to *all* the databases $\mathcal{D}$ *compatible* with $(\mathcal{O}, \mathcal{D})$, i.e., the databases $\mathcal{D}'$ extending $\mathcal{D}$ which "comply with" $\mathcal{O}$, i.e., such that its derived interpretation $\mathcal{I}(\mathcal{D}')$ is a model of $\mathcal{O}$.

Proposition 3.2.5 ([CdV$^+$06])**.** *For each UCQ φ of distinguished variables $\bar{x}$, every knowledge base $(\mathcal{O}, \mathcal{D})$ and every sequence of constants $\bar{c}$,*

$$(\mathcal{O}, \mathcal{D}) \models \varphi\{\bar{x} \mapsto \bar{c}\} \quad \textit{iff} \quad \bar{c} \in \bigcap_{\substack{\mathcal{D} \subseteq \mathcal{D}' \\ \mathcal{I}(\mathcal{D}') \models \mathcal{O}}} \textit{ans}(\varphi, \mathcal{D}').$$

By definition UCQs are contained in the positive existential fragment of $\mathbf{FO}$, $\mathbf{FO}^+_\exists$. Adding negation to UCQs gives rise to the class of $\mathbf{FO}$ queries. Hence, core SQL (SQL without aggregations) is syntactic sugar for $\mathbf{FO}$ and $\mathbf{FO}$ queries and its `SELECT-PROJECT-JOIN-UNION` fragment for $\mathbf{FO}^+_\exists$. Tree-shaped queries are defined as a proper fragment of UCQs and thus inherit their properties.

Definition 3.2.6 (Tree-shaped queries and their unions)**.** A *tree-shaped query* $\varphi(x)$ is a CQ with one distinguished variable x, called *root*, defined inductively by

$$\begin{aligned} \varphi(x) &\to A(x) \mid \exists y R(x,y) \mid \exists y R(x,y) \wedge \varphi(y) \mid \varphi(x) \wedge \varphi'(x) \\ R(x,y) &\to r(x,y) \mid r(y,x) \end{aligned} \tag{TCQ}$$

A *union of tree-shaped queries* is a union

$$\varphi(x) := \varphi_1(x) \vee \cdots \vee \varphi_k(x) \tag{UTCQ}$$

of TCQs. TCQs and UTCQs are said to be *boolean* when they contain no free variables.

Every TCQ $\varphi(x)$ rooted in x can be (bijectively) mapped to a directed adorned tree T_φ: each atom $r(z, z')$ gives rise to two nodes z and z' and an edge (z, z') with tag r and each atom $A(z)$ to tag A over node z [GHLS07, HT02].

3.3 Reasoning Problems

An OBDASs has to fulfil two key tasks: *(i)* it has to access the information its database stores, viz., compute the certain answers of a conjunctive query, and *(ii)* it has to be able to check whether any update violates an ontology constraint. Inspired by Vardi in [Var82] we are interested in knowing how difficult these tasks are w.r.t. the *size of the data*, a.k.a. *data complexity*. This is motivated by practical reasons. Relational databases may be very large. They may store terabytes of data. Queries and ontologies, on the other hand, tend to be comparatively small, often of negligible size compared to the sheer number of records in the tables to be joined, filtered and projected away to the answers, or updated. Thus, the main requirement of these systems is that they *scale to the data*. It is also of interest to consider the performance of the system w.r.t. all its inputs. To measure their scalability it is customary to consider the computational complexity of the decision problems associated to tasks *(i)* and *(ii)*, namely, knowledge base query answering and knowledge base satisfiability:

Definition 3.3.1 (Knowledge base query answering)**.** The *query answering* (KBQA) decision problem for UCQs and knowledge bases is the $\mathbf{FO}$ entailment problem stated as follows:

- **Input(s):** a knowledge base $(\mathcal{O}, \mathcal{D})$, a UCQ φ of distinguished variables $\bar{x}$ and a sequence $\bar{c}$ of $|\bar{x}|$ constants.
- **Question:** does there exist a substitution $\sigma(\cdot)$ s.t. *(i)* $\sigma(\bar{x}) = \bar{c}$ and *(ii)* $(\mathcal{O}, \mathcal{D}) \models \varphi\sigma$?

Definition 3.3.2 (Knowledge base satisfiability)**.** The knowledge base *satisfiability* (KBSAT) decision problem for knowledge bases is the $\mathbf{FO}$ satisfiability problem stated as follows:

- **Input(s):** a knowledge base $(\mathcal{O}, \mathcal{D})$.
- **Question:** Is $(\mathcal{O}, \mathcal{D})$ satisfiable?

Formally, the *data complexity* of KBQA and KBSAT arises when we consider $\mathcal{D}$ as the only input of the problem(s) [Var82]. When all the inputs of these reasoning problems are considered, we speak about their *combined complexity*.

Optimal data complexity is reached when KBQA and KBSAT are in **LSpace**, which is the complexity of relational database query evaluation. Indeed, such optimum is achieved when a log-space (in the data) reduction, known as *perfect rewriting* from KBQA and KBSAT to relational database query evaluation exists [CdL$^+$06, CdV$^+$06, CdL$^+$05a].

Definition 3.3.3 (Database query answering)**.** The *query answering* (QA) decision problem for UCQs and databases is the $\mathbf{FO}$ model checking problem stated as follows:

- **Input(s):** a database $\mathcal{D}$, a UCQ φ of distinguished variables $\bar{x}$ and sequence $\bar{c}$ of $|\bar{x}|$ constants.
- **Question:** does there exist an assignment $\gamma(\cdot)$ s.t. *(i)* $\gamma(\bar{x}) = \bar{c}$ and *(ii)* $\mathcal{I}(\mathcal{D}), \gamma \models \varphi$?

Theorem 3.3.4. *The problem of* QA *is:*

1. *in* **LSpace** *w.r.t. data complexity for UCQs and*
2. **NPTime**-*complete w.r.t. combined complexity for (U)CQs.*

Proof. Vardi shows in [Var82], that answering arbitrary $\mathbf{FO}$ queries over databases is in **LSpace** in data complexity (by reduction to the $\mathbf{FO}$ model checking problem). On the other hand, it can be shown that QA for UCQs is polynomially equivalent to the *query equivalence* problem in which we check whether, for any two queries φ and φ and every database $\mathcal{D}$, $\mathit{ans}(\varphi, \mathcal{D}) = \mathit{ans}(\varphi', \mathcal{D})$. The query equivalence problem for UCQs is known to be **NPTime**-complete (see [CM77]). □

Given the semantics of OBDAS, both data and combined complexity are influenced by the constructors allowed or dissallowed in the ontology and query languages supported by the system. In particular, full boolean conjunction and negation in the ontology and/or the query language give rise, in general, to **coNPTime**-hard data complexity for KBQA. The logics of the *DL-Lite* family, not being closed by neither construct, exhibit optimal data complexity, while $\mathcal{ELI}$ is still scalable (recall that it lacks negation), i.e., in **PTime**. The description logic $\mathcal{ALCI}$ (and $\mathcal{ALCHQI}$), since closed under negation and conjunction, is on the other hand, **coNPTime**-hard in data complexity. These results are formalized in the theorems below.

Theorem 3.3.5 ([CdV$^+$06, OCE08, LK07, Ros07])**.** *The problem of* KBQA *is:*

1. *in* **LSpace** *w.r.t. data complexity for UCQs and DL-Lite, DL-Lite$_R$, DL-Lite$_F$ and DL-Lite$_{R,\sqcap}$ ontologies;*
2. **PTime**-*complete w.r.t. data complexity for U(T)CQs and $\mathcal{ELI}$ ontologies;*
3. **coNPTime**-*complete w.r.t. data complexity for (T)CQs and $\mathcal{ALCI}$ and/or $\mathcal{ALCHQI}$ ontologies;*
4. **NPTime**-*hard w.r.t. combined complexity for CQs and $\mathcal{ELI}$ ontologies; and*
5. 2-**ExpTime**-*complete w.r.t. combined complexity for (U)CQs and $\mathcal{ALCI}$ and $\mathcal{ALCHQI}$ ontologies.*

Theorem 3.3.6 ([CdV$^+$06, BCN$^+$03])**.** *The problem of* KBSAT *is:*

1. *in* **LSpace** *w.r.t. data complexity for knowledge bases from DL-Lite, DL-Lite$_R$, DL-Lite$_F$ and DL-Lite$_{R,\sqcap}$; and*
2. **ExpTime**-*complete w.r.t. combined complexity for $\mathcal{ALCI}$ and $\mathcal{ALCHQI}$ knowledge bases.*

3.4 Related Formalisms

Ontology and query languages are thus closely related to (more precisely, contained in) a certain number of fragments of $\mathbf{FO}$, which possess interesting properties. In particular, the notions of knowledge base and queries can be generalized to $\mathbf{FO}$: ontologies can be seen as $\mathbf{FO}$ axiomatics, whereas queries are specific $\mathbf{FO}$ formulas.

A *restricted k-variable* fragment of $\mathbf{FO}$, denoted $\mathbf{FO}^k$, is any fragment constituted of all the $\mathbf{FO}$ formulas built using only k variables. In particular, the 2-variable fragment of $\mathbf{FO}$ [GKV97], $\mathbf{FO}^2$, is the fragment constructed using only two variables. The the extension $\mathbf{C}$ of $\mathbf{FO}$ with *counting quantifiers* is defined as the smallest set of formulas containing $\mathbf{FO}$ and such that the formula

$$\exists_{\leq k} x\, \varphi(x),$$

where $\varphi(x)$ is in $\mathbf{C}$, is in $\mathbf{C}$. Let $S(\bar{x})$ denote an arbitrary $\mathbf{FO}$ relational atomic formula of arity $|\bar{x}|$. The formulas of $\mathbf{GF}$, the *guarded* fragment of $\mathbf{FO}$ [Grä99] are defined inductively by

$$\varphi \rightarrow S(\bar{x}) \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \exists \bar{x}(S(\bar{x}) \wedge \varphi),$$

if $\bar{x} \subseteq FV(\varphi)$. The formulas of $\mathbf{FO}^+_\exists$, the *positive existential* fragment of $\mathbf{FO}$ are defined inductively by

$$\varphi \rightarrow S(\bar{x}) \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \exists \bar{x}\, \varphi.$$

Combining together these fragments gives rise to other interesting fragments. In particular, *(i)* $\mathbf{GF}^2$ the 2-variable guarded fragment of $\mathbf{FO}$, defined by

$$\mathbf{GF}^2 := \mathbf{FO}^2 \cap \mathbf{GF},$$

(ii) $\mathbf{C}^2$, the 2-variable fragment of $\mathbf{FO}$ with counting quantifiers, defined by

$$\mathbf{C}^2 := \mathbf{FO}^2 \cap \mathbf{C},$$

and *(ii)* $\mathbf{GC}^2$, the 2-variable guarded fragment of $\mathbf{FO}$ with counting quantifiers, defined by

$$\mathbf{GC}^2 := \mathbf{C}^2 \cap \mathbf{GF}.$$

The $\mathbf{FO}^2$, $\mathbf{GF}$ and $\mathbf{GF}^2$ fragments defined above are known to be decidable for SAT [GKV97, Grä99]. An important property of $\mathbf{FO}^+_\exists$ (and hence of UCQs) is that its formulas are closed under homomorphisms among interpretations.

Definition 3.4.1 (Homomorphism)**.** An *homomorphism* h among two interpretations $\mathcal{I}$ and $\mathcal{I}'$, denoted $\mathcal{I} \hookrightarrow_h \mathcal{I}'$, is a function $h\colon \mathbb{D}_\mathcal{I} \to \mathbb{D}_{\mathcal{I}'}$ s.t. for all relation symbols R of arity n and all $(c_1, \ldots, c_n) \in \mathbb{D}^n_\mathcal{I}$, if $(c_1, \ldots, c_n) \in S^\mathcal{I}$, then $(h(c_1), \ldots, h(c_n)) \in S^{\mathcal{I}'}$.

If the converse also holds, then we say that two interpretations $\mathcal{I}$ and $\mathcal{I}'$ are *homomorphically equivalent*, and write $\mathcal{I} \sim_h \mathcal{I}'$. A $\mathbf{FO}$ formula φ is said to be *closed under homomorphisms* whenever, for all interpretations $\mathcal{I}$ and $\mathcal{I}'$ if $\mathcal{I} \models \varphi$ and $\mathcal{I} \hookrightarrow_h \mathcal{I}'$, then $\mathcal{I}' \models \varphi$.

Theorem 3.4.2 ([CK90], Exercise 2.1.3)**.** *$\mathbf{FO}^+_\exists$ formulas, and a fortiori UCQs, are closed under homomorphisms.*

The database (**Ans**) and certain answers (**Cert**) semantics for UCQs that we introduced, since based on $\mathbf{FO}$ semantics, generalizes to $\mathbf{FO}$ queries and arbitrary $\mathbf{FO}$ knowledge bases. Clearly, Proposition 3.2.5 also generalizes to $\mathbf{FO}$ queries and arbitrary $\mathbf{FO}$ knowledge bases. This gives rise to a certain number of formal results that we will apply repeatedly in this thesis.

Theorem 3.4.3 ([PH08b])**.** *The data complexity of* KBQA *is* **coNPTime***-complete for $\mathbf{GC}^2$ and $\mathbf{FO}^+_\exists$ queries.*

Theorem 3.4.4 ([PH08b], Theorem 1)**.** *The data complexity of* KBSAT *is* **NPTime***-complete for $\mathbf{C}^2$.*

Since UTCQs are $\mathbf{C}^2$ formulas, this implies:

Corollary 3.4.5. *The data complexity of* KBQA *is* **coNPTime***-complete for $\mathbf{C}^2$ and (U)TCQs.*

These computational properties are inherited by the fragments they subsume.

3.5 Expressing Ontologies and Queries with Controlled English

In Chapter 3 we said that controlled languages (and in general, any fragment of a natural language) express, modulo compositionality, compositional translations $\tau(\cdot)$ and semantically enriched grammars, logic fragments, namely, the set of their ($\mathbf{FO}$ and $\mathbf{HO}$) meaning representations. Thereafter, their semantic complexity can be studied.

In this thesis we intend to engage in a fine-grained analysis of the data complexity of controlled English for data access. We would like, moreover, to make use of the wealth of proof techniques and results related to formal ontology and query languages to fullfill this aim. To this end, we propose to express in controlled English ontology and query languages:

Definition 3.5.1 (Expressing Query Languages)**.** Given a query language $\mathcal{Q}$, to *express $\mathcal{Q}$ in controlled language*, define a semantically enriched grammar $G_\mathcal{Q}$ of compositional translation $\tau_\mathcal{Q}(\cdot)$ defining an interrogative controlled language $L(G_\mathcal{Q})$ s.t. $\tau_\mathcal{Q}(L(G_\mathcal{Q})) = \mathcal{Q}$.

Definition 3.5.2 (Expressing Ontology Languages)**.** Given an ontology language $\mathcal{L}$, to *express* $\mathcal{L}$ *in controlled language*, define a semantically enriched grammar $G_{\mathcal{L}}$ of compositional translation $\tau_{\mathcal{O}}(\cdot)$ defining a declarative controlled language $L(G_{\mathcal{L}})$ s.t. $\tau_{\mathcal{O}}(L(G_{\mathcal{L}})) = \mathcal{L}$.

Clearly, if a controlled language or a pair of controlled languages, one declarative, $L_{\mathcal{L}}$, one interrogative $L_{\mathcal{Q}}$, express an ontology language **L** and a query language $\mathcal{Q}$ then, the semantic complexity of $L_{\mathcal{L}}$ and $L_{\mathcal{Q}}$ will *coincide* with the (set of) computational properties of **L** and $\mathcal{Q}$. In particular, this analysis can be done *construct by construct*, i.e., by examining the different combinations of function and content words covered by the controlled languages, insofar as their semantics will be captured exactly, modulo $\tau(\cdot)$, by the ontology and query language constructors into which they translate (as we saw in Chapter 3).

3.6 Summary

In this chapter we have overviewed the syntax, the semantics and the main (known) computational properties of the ontology and query languages that we will study in this thesis. On the one hand the following fragments of $\mathcal{ALCHQI}$: the *DL-Lite* family, $\mathcal{ALCI}$ and $\mathcal{ELI}$. On the other hand, as formal queries, (U)TCQs and (U)CQs. We have also recalled the decision problems we want to focus on: KBSAT (knowledge base satisfiability) and KBQA (knowledge base query answering), emphasizing why, in the context of OBDASs, we must focus on their data complexity. Our purpose is to express these ontology and query languages later in this thesis with several controlled languages. We argue that, modulo compositionality, this allows to provide a fine-grained analysis of their scalability to data based on the different combinations of function and content words they cover.

Chapter 4

Expressing *DL-Lite* and Tree-Shaped Queries

In this chapter we show two main results: *(i)* We express in controlled language the members of the *DL-Lite* family of description logics discussed and highlighted in Chapter 4. To express these description logics we define a declarative controlled language, Lite-English, expressing *DL-Lite*$_{\sqcap}$, which we later extend to express functionality and role assertions, i.e., to express *DL-Lite*$_{F,\sqcap}$ and *DL-Lite*$_{R,\sqcap}$. Since *DL-Lite*$_{\sqcap}$ extends *DL-Lite*$_{core}$, *DL-Lite*$_{F,\sqcap}$ extends both *DL-Lite*$_{F}$ and *DL-Lite*$_{\sqcap}$, and *DL-Lite*$_{R,\sqcap}$ extends both *DL-Lite*$_{R}$ and *DL-Lite*$_{\sqcap}$ (recall Figure 3.1 from Chapter 4), Lite-English expresses these description logics as well[1]. *(ii)* We express in controlled language TCQs. To express TCQs we define the interrogative controlled language GCQ-English, that expresses *graph-shaped conjunctive queries* (GCQs), a slight extension of TCQs which allows a restricted kind of loop in the tree-structure of TCQs.

Two reasons guide the choice of GCQs. On the one hand, we want to express in controlled language an optimal case of KBQA: answering CQs (a query language strictly more expressive than either TCQs or GCQs) over knowledge bases expressed in some *DL-Lite* description logic is in **LSpace** (recall Theorem 3.3.5 from Chapter 4). On the other hand, we want the declarations and questions of our fragments to remain close to grammatically correct English declarations and questions.

In particular, expressing the restricted loops allows a restricted coverage of English anaphoric pronouns by GCQ-English. Such coverage represents a trade-off between expressiveness and simplicity. CQs use arbitrary many variables, which, in formal semantic theory, correspond to anaphoric pronouns. However, we do not want to burden the casual user with arbitrarily long co-reference chains (which are in general difficult for speakers to keep in mind).

We also characterize the *relative and absolute expressive power* of Lite-English, by comparing the expressive power of the *DL-Lite* family and Pratt and Third's fragments of English [PHT06]. In particular, since SAT is tractable in combined complexity for the *DL-Lite* logics (SAT reduces to database query evaluation, see [CGL^{+}07]), we consider the tractable fragments of English, viz., COP, COP+TV and COP+TV+DTV. We exhibit a number of *closure properties* of the *DL-Lite* family (simulations, closure under unions of chains) and *logic embeddings* among the *DL-Lite* family and the fragments of English to establish, respectively, absolute and relative expressive power. This strategy allows us to pinpoint thereafter the controlled language constructs we gain with Lite-English, i.e., the English function words occurring in a controlled or heavily restricted form in Lite-English (negation, relatives) that give rise to intractability when used without restrictions in Pratt and Third's fragments.

The results on expressive power were first published in [BCT07a]. The results on GCQs and GCQ-English were first published in [Tho08].

[1]The work on Lite-English stems from joint work with R. Bernardi and D. Calvanese in [BCT07b].

4.1 Lite-English

In this section we present the syntax and vocabulary of Lite-English. Lite-English aims at expressing firstly the description logic *DL-Lite*$_\sqcap$ and, secondly, by means of suitable extensions, *DL-Lite*$_{R,\sqcap}$ and *DL-Lite*$_{F,\sqcap}$, and thus all the other description logics of interest from the *DL-Lite* family of description logics, *DL-Lite*$_{core}$, *DL-Lite*$_R$ and *DL-Lite*$_F$ which the former three subsume.

As Lite-English meaning representations we will consider the **FO** counterparts $C_l^{t_{x,y}}$ and $C_r^{t_{x,y}}$ of *DL-Lite* left and right concepts C_l and C_r rather than the concepts themselves, and call them *left* and *right formulas* and/or concepts. Similarly, we will use the binary atom $R^{t_{x,y}}$ to denote the role R. This makes sense since as we saw in the previous chapter, modulo the translations $\cdot^{t_{x,y}}$ and $\cdot^{t_{y,x}}$ the logics of the *DL-Lite* family can be seen as fragments of **FO**. Accordingly, we will consider

- instead of concept inclusion assertions $C_l \sqsubseteq C_r$, sentences $\forall x(C_l^{t_{x,y}} \Rightarrow C_r^{t_{x,y}})$,
- instead of role inclusion assertions $R \sqsubseteq R'$, $\forall x \forall y(R^{t_{x,y}} \Rightarrow R'^{t_{x,y}})$, and
- instead of functionality assertions $(\mathit{funct}\ R)$, $\forall x(\top^{t_{x,y}} \Rightarrow \exists_{\leq 1} R^{t_{x,y}} \wedge \top^{t_{y,x}})$.

Please notice that since types and typed lambda expressions are not strictly necessary for expressing *DL-Lite*$_\sqcap$, we will omit any explicit mention of the them in the definition of Lite-English and of the compositional translation $\tau(\cdot)$. To simplify the proofs that show that a controlled language expresses an ontology language and/or a query language, we introduce the notion of *structural equivalence*:

Definition 4.1.1 (Structural equivalence)**.** A **HO** formula $\psi := \lambda x_1 \cdots \lambda x_n.\chi$ is said to be *structurally equivalent* to a **FO** formula φ with n free variables, in symbols $\varphi \equiv_s \psi$, whenever $\chi \equiv \varphi$, i.e., whenever χ and φ are **FO** equivalent (i.e., whenever $\mathit{Mod}(\chi) = \mathit{Mod}(\varphi)$).

4.1.1 Expressing *DL-Lite*$_\sqcap$

Lite-English is defined by a semantically enriched context-free grammar whose phrase structure rules are shown in Figure 4.1 and whose content and function lexicon are shown in Figure 4.2. We consider as function words: pronouns (e.g., “somebody”), determiners (e.g. “every”), conjunctions (e.g. “and”), etc., which express, ultimately, *DL-Lite*$_\sqcap$ logical operators in **FO** format. As content words we consider: common nouns (e.g. “man”), proper nouns (e.g. “Julian”), attributive and qualificative (a.k.a. intersective) adjectives (e.g., “brave”), intransitive and transitive verbs (e.g., “leaves” or “loves”). We do not consider ditransitive verbs (i.e., we exclude verbs like “gives” from our lexicon).

Proper nouns stand for individuals, common nouns for *DL-Lite*$_\sqcap$ atomic concepts, adjectives for attributes, and recursive set-typed constituents (verb phrases and nominals), for arbitrary *DL-Lite*$_\sqcap$ concepts.

We *subcategorize* syntactic categories into *left* and *right* categories (by means of the indexes l and r, respectively), to capture the distinction made in *DL-Lite*$_\sqcap$ among left and right concepts. Thus, our grammar contains two separate sets of phrase structure rules, one defining left constituents, which express left concepts C_l, and another that defines right constituents, which express right concepts C_r.

Inverted roles can be expressed by considering their passive forms. For simplicity, however, we will disregard all morphosyntactic issues, which are, anyway, easy to deal with. We will consider declarative sentences inflected in the third person, of masculine gender and singular number, and in present tense and active voice.

Lemma 4.1.2. *For every sentence D of Lite-English, there exists an assertion α of DL-Lite*$_\sqcap$ *s.t.* $\tau(D) \equiv_s \alpha$.

(Phrase structure rules)	(Semantic actions)
$\mathbf{S}_O \to \mathbf{NP}_l\, \mathbf{VP}_r$	$\tau(\mathbf{S}_O) := \tau(\mathbf{NP}_l)(\tau(\mathbf{VP}_r))$
$\mathbf{S}_g^l \to \mathbf{VP}_l$	$\tau(\mathbf{S}_g^l) := \tau(\mathbf{VP}_l)$
$\mathbf{S}_g^r \to \mathbf{VP}_r$	$\tau(\mathbf{S}_g^r) := \tau(\mathbf{VP}_r)$
$\mathbf{N}_l \to \mathbf{N}_l\, \mathbf{RelC}_l$	$\tau(\mathbf{N}_l) := \tau(\mathbf{RelC}_l)(\tau(\mathbf{N}_l))$
$\mathbf{N}_l \to \mathbf{N}$	$\tau(\mathbf{N}_l) := \tau(\mathbf{N})$
$\mathbf{N}_l \to \mathbf{Adj}\, \mathbf{N}_l$	$\tau(\mathbf{N}_l) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_l))$
$\mathbf{N}_r \to \mathbf{N}_r\, \mathbf{RelC}_r$	$\tau(\mathbf{N}_r) := \tau(\mathbf{RelC}_r)(\tau(\mathbf{N}_r))$
$\mathbf{N}_r \to \mathbf{N}$	$\tau(\mathbf{N}_r) := \tau(\mathbf{N})$
$\mathbf{N}_r \to \mathbf{Adj}\, \mathbf{N}_r$	$\tau(\mathbf{N}_r) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_r))$
$\mathbf{VP}_l \to \mathbf{TV}\, \mathbf{NP}_l$	$\tau(\mathbf{VP}_l) := \tau(\mathbf{NP}_l)(\tau(\mathbf{TV}))$
$\mathbf{VP}_l \to \mathbf{IV}$	$\tau(\mathbf{VP}_l) := \tau(\mathbf{IV})$
$\mathbf{VP}_l \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_l) := \tau(\mathbf{Adj})$
$\mathbf{VP}_l \to$ is a $\mathbf{N}_l$	$\tau(\mathbf{VP}_l) := \tau(\mathbf{N}_l)$
$\mathbf{VP}_l \to \mathbf{VP}_l\, \mathbf{Crd}\, \mathbf{VP}_l$	$\tau(\mathbf{VP}_l) := (\tau(\mathbf{Crd})(\tau(\mathbf{VP}_l)))(\tau(\mathbf{VP}_l))$
$\mathbf{VP}_r \to \mathbf{VP}_r\, \mathbf{Crd}\, \mathbf{VP}_r$	$\tau(\mathbf{VP}_r) := (\tau(\mathbf{Crd})(\tau(\mathbf{VP}_r)))(\tau(\mathbf{VP}_r))$
$\mathbf{VP}_r \to \mathbf{TV}\, \mathbf{NP}_r$	$\tau(\mathbf{VP}_r) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_r))$
$\mathbf{VP}_r \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_r) := \tau(\mathbf{Adj})$
$\mathbf{VP}_r \to$ is a $\mathbf{N}_r$	$\tau(\mathbf{VP}_r) := \tau(\mathbf{N}_r)$
$\mathbf{VP}_r \to \mathbf{IV}$	$\tau(\mathbf{VP}_r) := \tau(\mathbf{IV})$
$\mathbf{NP}_l \to \mathbf{Pro}_l\, \mathbf{RelC}_l$	$\tau(\mathbf{NP}_l) := \tau(\mathbf{Pro}_l)(\tau(\mathbf{RelC}_l))$
$\mathbf{NP}_l \to \mathbf{Det}_l\, \mathbf{N}_l$	$\tau(\mathbf{NP}_l) := \tau(\mathbf{Det}_l)(\tau(\mathbf{N}_l))$
$\mathbf{NP}_r \to \mathbf{Det}_r\, \mathbf{N}_r$	$\tau(\mathbf{NP}_r) := \tau(\mathbf{Det}_r)(\tau(\mathbf{N}_r))$
$\mathbf{RelC}_l \to \mathbf{Relp}\, \mathbf{S}_g^l$	$\tau(\mathbf{RelC}_l) := \tau(\mathbf{Relp})(\tau(\mathbf{S}_g^l))$
$\mathbf{RelC}_r \to \mathbf{Relp}\, \mathbf{S}_g^r$	$\tau(\mathbf{RelC}_r) := \tau(\mathbf{Relp})(\tau(\mathbf{S}_g^r))$
$\mathbf{S}_D \to \mathbf{NP}_D\, \mathbf{VP}_D$	$\tau(\mathbf{S}_D) := \tau(\mathbf{NP}_D)(\tau(\mathbf{VP}_D))$
$\mathbf{VP}_D \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_D) := \tau(\mathbf{Adj})$
$\mathbf{VP}_D \to$ is a $\mathbf{N}$	$\tau(\mathbf{VP}_D) := \tau(\mathbf{N})$
$\mathbf{VP}_D \to \mathbf{TV}\, \mathbf{NP}_D$	$\tau(\mathbf{VP}_D) := \tau(\mathbf{TV})(\tau(\mathbf{NP}_D))$
$\mathbf{NP}_D \to \mathbf{Pn}$	$\tau(\mathbf{NP}_D) := \tau(\mathbf{Pn})$

Figure 4.1: Lite-English

(Function lexicon)

$\mathbf{Pro}_l \rightarrow$ anybody	$\tau(\mathbf{Pro}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x))$
$\mathbf{Det}_l \rightarrow$ every	$\tau(\mathbf{Det}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x))$
$\mathbf{Det}_l \rightarrow$ no	$\tau(\mathbf{Det}_l) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow \neg Q(x))$
$\mathbf{Pro}_l \rightarrow$ somebody	$\tau(\mathbf{Pro}_l) := \lambda P.\exists x P(x)$
$\mathbf{Pro}_r \rightarrow$ somebody	$\tau(\mathbf{Pro}_r) := \lambda P.\exists x P(x)$
$\mathbf{Crd} \rightarrow$ and	$\tau(\mathbf{Crd}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$\mathbf{Relp} \rightarrow$ who	$\tau(\mathbf{Relp}) := \lambda P.\lambda x.P(x)$
$\mathbf{Relp} \rightarrow$ that	$\tau(\mathbf{Relp}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$

(Content lexicon)

$\mathbf{N} \rightarrow$ man	$\tau(\mathbf{N}) := \lambda x.\mathit{Man}(x)$
$\mathbf{N} \rightarrow$ woman	$\tau(\mathbf{N}) := \lambda x.\mathit{Woman}(x)$
	$\vdots$
$\mathbf{TV} \rightarrow$ attacks	$\tau(\mathbf{TV}) := \lambda\beta.\lambda x.\beta(\lambda y.\mathit{attacks}(x, y))$
$\mathbf{TV} \rightarrow$ loves	$\tau(\mathbf{TV}) := \lambda\beta.\lambda x.\beta(\lambda y.\mathit{loves}(x, y))$
$\mathbf{TV} \rightarrow$ likes	$\tau(\mathbf{TV}) := \lambda\beta.\lambda x.\beta(\lambda y.\mathit{likes}(x, y))$
	$\vdots$
$\mathbf{IV} \rightarrow$ runs	$\tau(\mathbf{IV}) := \lambda x.\mathit{Run}(x)$
	$\vdots$
$\mathbf{Pn} \rightarrow$ Julian	$\tau(\mathbf{Pn}) := \lambda P.P(\text{Julian})$
$\mathbf{Pn} \rightarrow$ Persia	$\tau(\mathbf{Pn}) := \lambda P.P(\text{Persia})$
	$\vdots$

Figure 4.2: Lite-English

Proof. To prove this result we need to prove that

$$\begin{array}{c}\text{for each } \mathbf{VP}_f \text{ or } \mathbf{N}_f \text{ constituent, for } f \in \{l, r\}, \text{ there exists} \\ \text{a concept } C_f(x) \text{ s.t. } \tau(\mathbf{VP}_f) \equiv_s C_f(x) \text{ or } \tau(\mathbf{N}_f) \equiv_s C_f(x)\end{array} \quad (\dagger)$$

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

- ($n = 1$) There are five 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 four cases $\tau(\cdot)$ maps them to $\lambda x.A(x)$, where $A(x)$ is a concept name or atomic formula.
- ($n = k + 1$) We look at a couple of cases only. The argument is similar for all the remaining cases.
  - $\mathbf{VP}_l \Longrightarrow^{k+1} \mathbf{VP}'_l\,\mathbf{Crd}\,\mathbf{VP}''_l$. Then $\mathbf{VP}'_l \Longrightarrow^k w'$ and $\mathbf{VP}''_l \Longrightarrow^k w''$, for some sequences w', w'' of terminals and non-terminals. By IH there exist $C'_l(x)$ and $C''_l(x)$ s.t. $\tau(\mathbf{VP}'_l) \equiv_s C'_l(x)$ and $\tau(\mathbf{VP}''_l) \equiv_s C''_l(x)$. Whence:

$$\begin{array}{rl}\tau(\mathbf{VP}_l) & =_{df} \tau(\mathbf{Crd})(\tau(\mathbf{VP}'_l))(\tau(\mathbf{VP}''_l)) \\ & =_{ih} \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)(\lambda z.C'_l(z))(\lambda w.C''_l(w)) \\ & \rhd\ \lambda x.(C'_l(x) \wedge C''_l(x)),\end{array}$$

    which is structurally equivalent to the left concept $C_l(x) = C'_l(x) \wedge C''_l(x)$.
  - $\mathbf{N}_r \Longrightarrow^{k+1} \mathbf{N}_r\ \mathbf{RelC}_r \Longrightarrow^k \mathbf{N}\ \mathbf{Relp}\ \mathbf{S}^r_g$. Now $\mathbf{S}^r_g \Longrightarrow \mathbf{VP}_r$ and $\mathbf{VP}_r \Longrightarrow^{k-1} w''$ for some sequence w'' of terminals and non-terminals. By IH on derivations of length $\leq k$ rooted in $\mathbf{VP}_r$, $\mathbf{VP}_r \equiv_s C_r(x)$, for some right concept $C_r(x)$. On the other hand $\mathbf{N}_r \Longrightarrow^{k-1} w'$ with $\tau(\mathbf{N}_r) \equiv_s C'_r(x)$, again by IH. Therefore,

$$\begin{array}{rl}\tau(\mathbf{N}_r) & =_{df} \tau(\mathbf{Relp})(\tau(\mathbf{N}_r))(\tau(\mathbf{S}^r_g)) \\ & =_{ih} \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)(\lambda z.C'_r(z))(\lambda w.C''_r(w)) \\ & \rhd\ \lambda x.C'_r(x) \wedge C''_r(x),\end{array}$$

    and $\tau(\mathbf{N}_r) \equiv_s C'_r(x) \wedge C''_r(x)$, a right concept or formula.

We are now ready to associate a complete meaning representation to each Lite-English sentence D. We have two cases to consider

i. $\mathbf{S}_O \Longrightarrow^* \mathbf{NP}_l\ \mathbf{VP}_r \Longrightarrow^* \mathbf{Det}_l\ \mathbf{N}_l\ \mathbf{VP}_r$, and
ii. $\mathbf{S}_O \Longrightarrow^* \mathbf{NP}_l\ \mathbf{VP}_r \Longrightarrow^* \mathbf{Pro}_l\ \mathbf{RelC}_l\ \mathbf{VP}_r$.

In both cases, modulo (†) is is easy to see that $\tau(\mathbf{S}_O) \equiv_s \forall x(C_l(x) \Rightarrow C_r(x))$. □

Lemma 4.1.3. *Every DL-Lite$_\sqcap$ assertion α is the image by $\tau(\cdot)$ of a sentence S of Lite-English up to structural equivalence.*

Proof. To prove this result we need to prove that

$$\begin{array}{c}\text{for each DL-Lite}_\sqcap\ C_f(x) \text{ concept, there exists a Lite-English } \mathbf{VP}_f \\ \text{or } \mathbf{N}_f \text{ s.t. } \tau(\mathbf{VP}_f) \equiv_s C_f \text{ or } \tau(\mathbf{N}_f) \equiv_s C_f, \text{ for } f \in \{l, r\}\end{array} \quad (\dagger)$$

by (a tedious, albeit simple) structural induction on concepts $C_f(x)$. We do it only for left concepts. The proof for right concepts proceeds similarly.

- **(Basis)** There are two possibilities. Either $C_l(x) := A(x)$, 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 $C_l(x) := \exists y R(x, y)$ and we consider: $\mathbf{VP}_l \Longrightarrow^* \mathbf{TV}\ \mathbf{NP}_l \Longrightarrow^* \mathbf{TV}\ \mathbf{Pro}_l \Longrightarrow^* R$s somebody.

– **(Inductive step)** $C_l(x) := C'_l(x) \wedge C''_l(x)$. By IH there exists a $\mathbf{VP}'_l$ and a $\mathbf{VP''}_l$ (resp. a $\mathbf{N}'_l$ and a $\mathbf{N''}_l$) s.t. $\mathbf{VP}'_l \Longrightarrow^* w'$ and $\mathbf{VP''}_l \Longrightarrow^* w''$ with, resp., $\tau(\mathbf{VP}'_l) \equiv_s C'_l(x)$ and $\tau(\mathbf{VP''}_l) \equiv_s C''_l(x)$. The desired $\mathbf{VP}_l$ (or $\mathbf{N}_l$) is given by $\mathbf{VP}_l \Longrightarrow^* \mathbf{VP}'_l\ \mathbf{Crd}\ \mathbf{VP''}_l \Longrightarrow^* w'$ and w'' of meaning representation $C'_l(x) \wedge C''_l(x)$.

We now need to show that when put together into assertions, they can be captured by some Lite-English sentence. Let $\alpha := \forall x(C_l(x) \Rightarrow \pm C_r(x))$ be a *DL-Lite*$_\sqcap$ assertion. There are two possibilities:

i. either $\mathbf{S}_O \Longrightarrow^* \mathbf{NP}_l\ \mathbf{VP}_r \Longrightarrow^* \mathbf{Det}_l\ \mathbf{N}_l\ \mathbf{VP}_r \Longrightarrow^*$ no/every $w\ w'$, or
ii. $\mathbf{S}_O \Longrightarrow^* \mathbf{NP}_l\ \mathbf{VP}_r \Longrightarrow^* \mathbf{Pro}_l\ \mathbf{Relp}_l\ \mathbf{S}^l_g\ \mathbf{VP}_r \Longrightarrow^* \mathbf{Pro}_l\,\mathbf{Relp}_l\,\mathbf{VP}_l\,\mathbf{VP}_r \Longrightarrow^*$ anybody who $\mathbf{VP}_l\,\mathbf{VP}_r \Longrightarrow^*$ anybody who $w\,w'$.

Modulo (†), this means that "no/every $\mathbf{N}_l\ \mathbf{VP}_r$" and "anybody who $\mathbf{VP}_l\ \mathbf{VP}_r$" are the desired Lite-English sentences (or sentence patterns). □

From Lemmas 4.1.2 and 4.1.3 we immediately derive:

Theorem 4.1.4 (Lite-English)**.** *Lite-English expresses DL-Lite*$_\sqcap$.

In general, Lite-English sentences respect the patterns

$$\mathbf{S}_O \rightarrow \mathbf{Det}_l\ \mathbf{N}_l\mathbf{VP}_r \qquad \text{and} \qquad \mathbf{S}_O \rightarrow \mathbf{Det}_l\ \mathbf{RelC}_l\ \mathbf{VP}_r$$

where $\mathbf{Det}_l\ \mathbf{N}_l$ (resp. $\mathbf{Det}_l\ \mathbf{RelC}_l$) corresponds to the *subject* of the sentence and $\mathbf{VP}_r$ to the predicate of the sentence. Subjects map to left $C_l(x)$ concepts and predicates to right $C_r(x)$ concepts. Such sentence patterns are ultimately mapped, modulo $\tau(\cdot)$ and structural equivalence, to $\mathbf{FO}$ sentences $\forall x(C_l(x) \Rightarrow C_r(x))$, which are the image by translations $\cdot^{t_{x,y}}$ and $\cdot^{t_{y,x}}$ of a *DL-Lite*$_\sqcap$ assertion $C_l \sqsubseteq C_r$.

For example, existential quantification, conveyed by the $\mathbf{Pro}_r$ and $\mathbf{Pro}_l$ "somebody" (and of meaning representation $\lambda P.\exists x P(x)$, i.e., a generalized quantifier [BC80]), can occur both in the subject $\mathbf{N}_l$ or $\mathbf{RelC}_l$ constituents or in the predicate $\mathbf{VP}_r$ constituent (and within their arbitrarily nested subordinated clauses). This is because *DL-Lite*$_\sqcap$ unqualified existential roles (in $\mathbf{FO}$: formulas like $\exists y\, r(x,y)$ with one free variable) can occur both to the left and to the right of $\Rightarrow$ (or of $\sqsubseteq$ in description logic notation), as seen in Chapter 4, Table 3.2.

Example 4.1.5. Figure 4.3 shows that

Every man loves somebody. (4.1)

is in Lite-English and expresses the assertion $\forall x(\mathit{Man}(x) \Rightarrow \exists y \mathit{loves}(x,y))$, which corresponds to the *DL-Lite*$_\sqcap$ ontology assertion $\mathit{Man} \sqsubseteq \exists \mathit{love}$. At each node, the meaning representation built is the reduct (by $\rhd$) of its immediate successors, down to the yield. Similiarly, Figure 4.4 shows that

Julian attacks Persia. (4.2)

is also recognized. As hinted above, the final logical assertion is attained through lambda calculus β-reduction. On the other hand, English sentences like

*Some man loves anybody. (4.3)

do not belong to Lite-English. Why? Because *(i)* "some" cannot occur in subject position, *(ii)* "anybody" cannot occur in predicate position. ♣

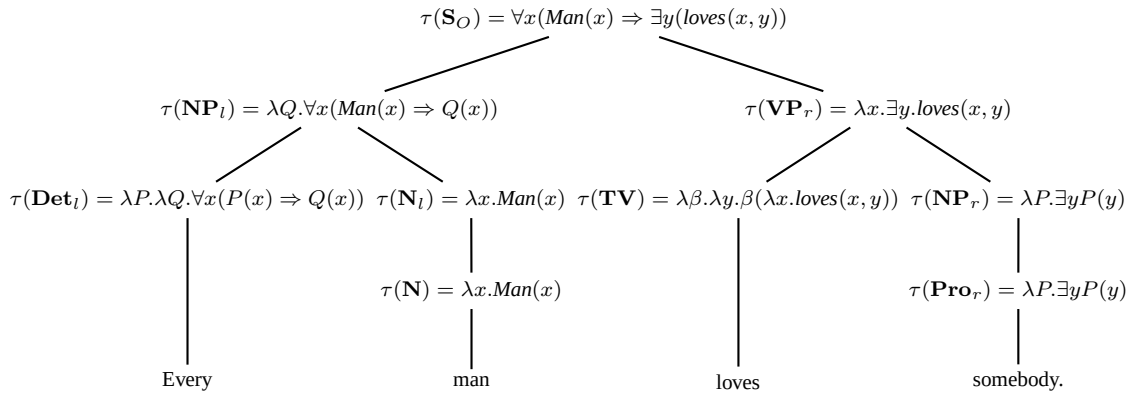


Figure 4.3: Parse tree for "Every man loves somebody".

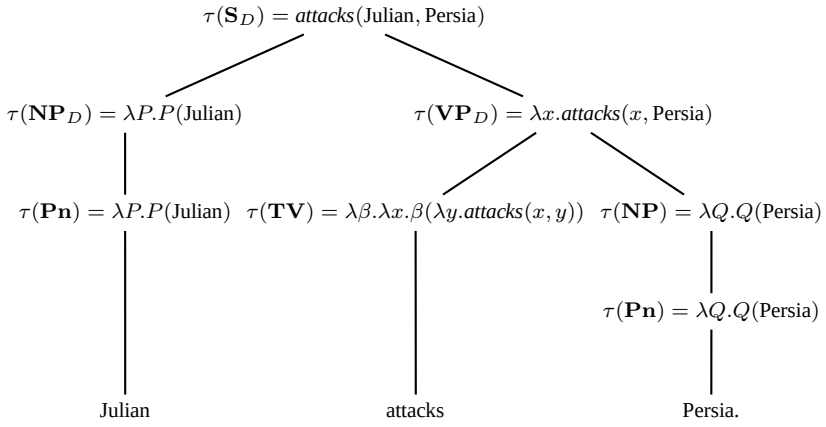


Figure 4.4: Parse tree for "Julian attacks Persia."

4.1.2 Expressing *DL-Lite*$_{F,\sqcap}$ and *DL-Lite*$_{R,\sqcap}$

To express the description logics *DL-Lite*$_{F,\sqcap}$ and *DL-Lite*$_{R,\sqcap}$ we enrich the grammar of Lite-English with two separate (i.e., disjoint) sets of ad hoc phrase structure rules designed to parse solely role inclusions and functionality assertions, obtaining Lite-English$_R$ and Lite-English$_F$. As *DL-Lite*$_{F,\sqcap}$ contains *DL-Lite*$_F$, *DL-Lite*$_{R,\sqcap}$ contains *DL-Lite*$_R$, and all, including *DL-Lite*$_\sqcap$, contain *DL-Lite*$_{core}$, we express all the *DL-Lite* logics that interest us. Figure 4.6 shows a sample Lite-English$_R$ parse tree, while Figure 4.5 shows a Lite-English$_F$ parse tree.

The extension of the grammar for Lite-English$_F$ is as follows:

(Phrase structure rules)	(Semantic actions)
$\mathbf{S}_F \rightarrow \mathbf{NP}^l_F\ \mathbf{VP}_F$	$\tau(\mathbf{S}_F) := \tau(\mathbf{NP}^l_F)(\tau(\mathbf{VP}_F))$
$\mathbf{NP}^l_F \rightarrow \mathbf{Det}^l_F\ \mathbf{N}_F$	$\tau(\mathbf{NP}^l_F) := \tau(\mathbf{Det}^l_F)(\tau(\mathbf{N}_F))$
$\mathbf{NP}^r_F \rightarrow \mathbf{Det}^r_F\ \mathbf{N}_F$	$\tau(\mathbf{NP}^r_F) := \tau(\mathbf{Det}^r_F)(\tau(\mathbf{N}_F))$
$\mathbf{VP}_F \rightarrow \mathbf{TV}\ \mathbf{NP}^r_F$	$\tau(\mathbf{VP}_F) := \tau(\mathbf{TV})(\tau(\mathbf{NP}^r_F))$

(Function lexicon)

$$\begin{array}{ll} \mathbf{Det}^r_F \to \text{every} & \tau(\mathbf{Det}^r_F) := \lambda P.\lambda Q \forall x(P(x) \Rightarrow Q(x)) \\ \mathbf{Det}^l_F \to \text{at most one} & \tau(\mathbf{Det}^l_F) := \lambda P.\lambda Q \exists_{\leq 1} x(P(x) \wedge Q(x)) \end{array}$$

(Content lexicon)

$$\mathbf{N}_F \to \text{thing} \qquad \tau(\mathbf{N}_F) := \lambda x.\top(x)$$

Theorem 4.1.6. *For each Lite-English$_F$ sentence D_F there exists a DL-Lite$_{F,\sqcap}$ assertion α_F s.t. $\tau(D_F) \equiv_s \alpha_F$. Conversely, each DL-Lite$_{F,\sqcap}$ assertion α_F is the image by $\tau(\cdot)$ of a sentence D_F of Lite-English$_F$.*

Proof. In neither case is there any induction to be made, since there are no recursive constituents. We just need to reason by cases considering all the possible (finite) combinations of phrase-structure grammar rules as we did when closing the proof of Lemmas 4.1.2 and 4.1.3. □

The extension of the grammar for Lite-English$_R$ is as follows:

(Phrase structure rules)	(Semantic actions)
$\mathbf{S}_R \to \mathbf{NP}^l_{i,j}\ \mathbf{VP}^R_{i,j}$	$\tau(\mathbf{S}_R) := \tau(\mathbf{NP}^l_{i,j})(\tau(\mathbf{VP}^R_{i,j}))$
$\mathbf{NP}^l_{i,j} \to \mathbf{Pro}^l_i\ \mathbf{RelC}^l_{i,j}$	$\tau(\mathbf{NP}^l_{i,j}) := \tau(\mathbf{Pro}^l_i)(\tau(\mathbf{RelC}^l_{i,j}))$
$\mathbf{RelC}^l_{i,j} \to \mathbf{Relp}^l_i\ \mathbf{S}^l_{g_i,j}$	$\tau(\mathbf{RelC}^l_{i,j}) := \tau(\mathbf{Relp}^l_i)(\tau(\mathbf{S}^l_{g_i,j}))$
$\mathbf{S}^l_{g_i,j} \to \mathbf{NP}_{g_i}\ \mathbf{VP}^r_{i,j}$	$\tau(\mathbf{S}_{g_i,j}) := \tau(\mathbf{NP}_{g_i})(\tau(\mathbf{VP}^r_{i,j}))$
$\mathbf{VP}^R_{i,j} \to \mathbf{TV}\ \mathbf{NP}^R_j$	$\tau(\mathbf{VP}^R_{i,j}) := \tau(\mathbf{TV})(\tau(\mathbf{NP}^R_j))$
$\mathbf{NP}^R_i \to \mathbf{Pro}^R_i$	$\tau(\mathbf{NP}^R_i) := \tau(\mathbf{Pro}^R_i)$

(Function lexicon)

$$\begin{array}{ll} \mathbf{Pro}^l_i \to \text{anybody} & \tau(\mathbf{Pro}^l_i) := \lambda P.\lambda Q.\forall x(P(x) \Rightarrow Q(x)) \\ \mathbf{Pro}^r_i \to \text{somebody} & \tau(\mathbf{Pro}^r_i) := \lambda P.\exists x P(x) \\ \mathbf{NP}^l_{g_i} \to t_i & \tau(\mathbf{NP}^l_{g_i}) := \lambda P.P(x) \\ \mathbf{Relp}^l_i \to \text{who} & \tau(\mathbf{Relp}^l_i) := \lambda P.\lambda x.P(x) \\ \mathbf{Pro}^R_i \to \text{him} & \tau(\mathbf{Pro}^R_i) := \lambda P.P(x) \end{array}$$

Theorem 4.1.7. *For each Lite-English$_R$ sentence D_R there exists a DL-Lite$_{R,\sqcap}$ assertion α_R s.t. $\tau(D_R) \equiv_s \alpha_R$. Conversely, each DL-Lite$_{R,\sqcap}$ assertion α_R is the image by $\tau(\cdot)$ of a sentence D_R of Lite-English$_R$.*

Proof. Again, no inductions are needed, since there are no recursive constituents. We just need to reason by cases considering all the possible (finite) combinations of phrase-structure grammar rules as we did when closing the proof of Lemmas 4.1.2 and 4.1.3. □

Remark 4.1.8. Notice that in figure 4.6, $\tau(\cdot)$ assigns to the subject $\mathbf{NP}^l_{i,j}$ constituent the meaning representation $\lambda P.\forall x \forall y(\textit{loves}(x,y) \Rightarrow P(x))$ rather than $\lambda P.\forall x(\exists y \textit{loves}(x,y) \Rightarrow P(x))$. However, since y does not occur free in $P(x)$, both expressions are logically equivalent. This proviso allows us to correctly generate a role inclusion from an input utterance.

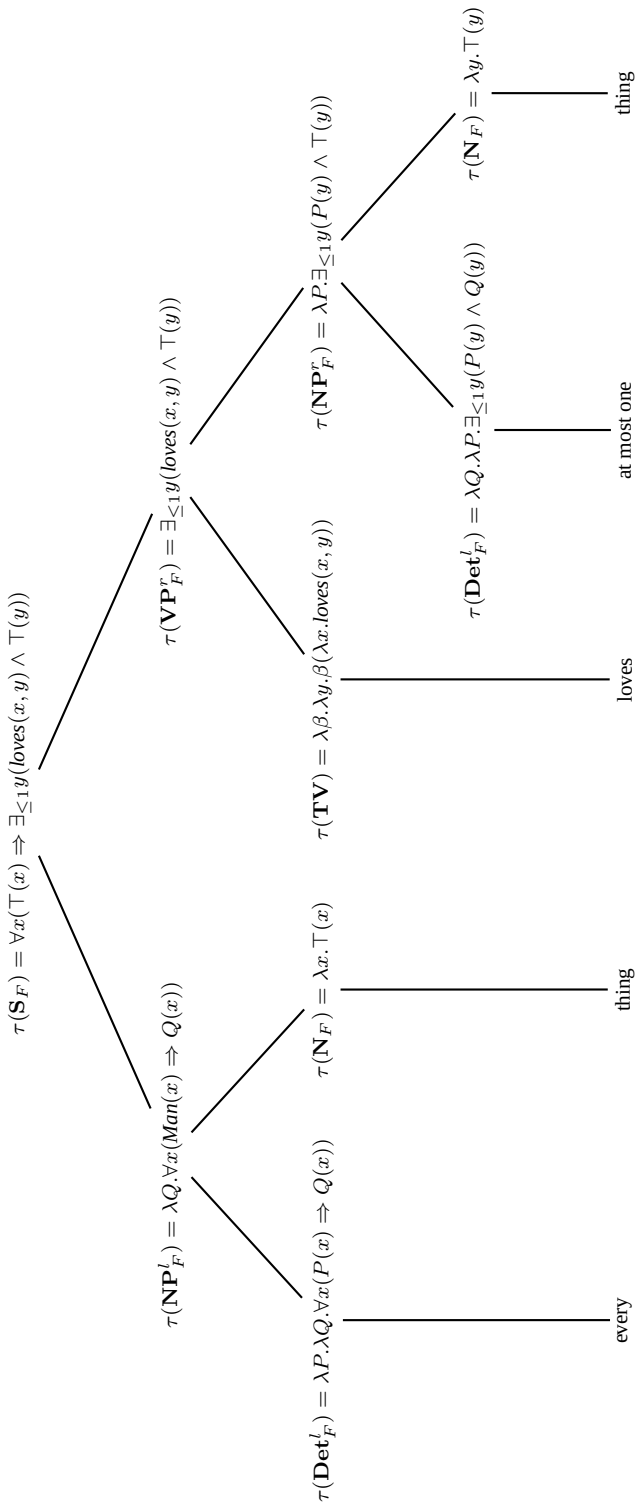


Figure 4.5: Expressing functionality assertions.

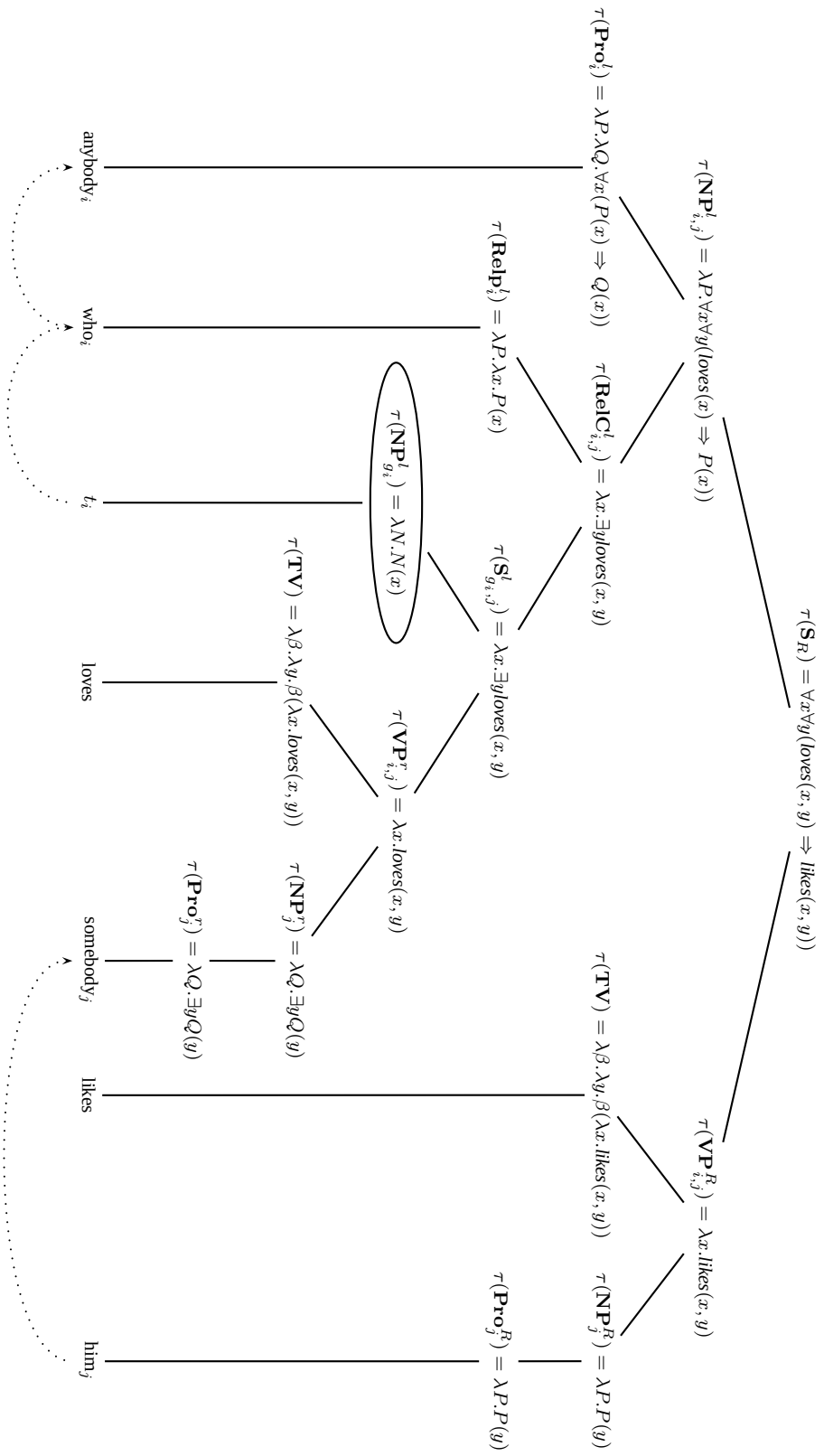


Figure 4.6: Expressing role inclusion.

4.1.3 Expressive Power of Lite-English

The (semantic) expressive power of a logic (in the restricted sense of a fragment of $\mathbf{FO}$ or $\mathbf{HO}$) consists in its model-theoretic properties. But there are alternative ways of defining expressive power. One can speak about *(i) absolute* and *(ii) relative* expressive power.

Absolute expressive power is conveyed by so-called *characterization* theorems that state *closure properties* that hold for their models, and provide a criterion for delimiting the logic's class of models.

Relative expressive power can be explored, on the other hand, by *simulating* a logic, viz., by defining a model-preserving logic embedding.

Modulo simulations, embedding logics inherit the properties of the embedded logic [Str05]. In this section we consider only fragments of $\mathbf{FO}$. Let **Sig** be a $\mathbf{FO}$ signature and let $For_{\mathbf{Sig}}$ denote the set of all $\mathbf{FO}$ formulas that can be constructed over **Sig**. A *logic* or *logic fragment* is then every $\mathbf{L} \subseteq For_{\mathbf{Sig}}$.

Definition 4.1.9 (Semantic expressiveness)**.** A *semantic property* is a class of interpretations. It said to be *expressible* by a logic $\mathbf{L}$ iff there exists a formula $\varphi \in \mathbf{L}$ such that its class of models $Mod(\varphi)$ coincides with this property. The *expressive power* of $\mathbf{L}$ is the union of all such expressible properties.

Definition 4.1.10 (Logic simulation)**.** Let $\mathbf{L}$, $\mathbf{L}'$ be two logics over signatures $\mathbf{Sig}$ and $\mathbf{Sig}$'. $\mathbf{L}'$ is said to be *at least as expressive as* $\mathbf{L}$, or, equivalently, $\mathbf{L}$ *is contained in* $\mathbf{L}'$, iff there exists a translation $\cdot^t$ from $For_{\mathbf{Sig}}$ to $For_{\mathbf{Sig}'}$ such that, for every $\varphi \in \mathbf{L}$, and every interpretation $\mathcal{I}$ over $\mathbf{Sig}$,

$$\mathcal{I} \models \varphi \quad \text{iff} \quad \mathcal{I} \models \varphi^t. \tag{\textbf{Sim}}$$

The translation $\cdot^t$ is called in such case a *model-preserving* translation, a *simulation* or by others, a *logic (homomorphic) embedding* [Str05]. It can be extended to sets of formulas in the usual way. The contains relation is trivially a pre-order (i.e., reflexive and transitive). Its symmetric closure, the *equally expressive as* relation, is therefore an equivalence relation.

Proposition 4.1.11. *Let* $\mathbf{L}$, $\mathbf{L}'$ *be two logics s.t.* $\mathbf{L}$ *is contained in* $\mathbf{L}'$. *Then every semantic property expressible in* $\mathbf{L}$ *is expressible in* $\mathbf{L}'$.

However, this does not preclude their expressive power from overlapping: logic $\mathbf{L}$ is said to *overlap* in expressive power with $\mathbf{L}'$ iff there exists a semantic property expressible by both $\mathbf{L}$ and $\mathbf{L}'$.

Expressive Power of $\mathbf{FO}$. In this section we introduce some model-theoretic properties of $\mathbf{FO}$ on which we leverage later. Among these, a closure property of $\mathbf{FO}$ $\forall^*\exists^*$-sentences: closure under unions of chains [2]. We follow in this section Cori and Lascar (see [CL03], Vol. 2, Chapter 8, Section 5.5).

Definition 4.1.12 (Sub-interpretation)**.** Given two interpretations $\mathcal{I}$ and $\mathcal{I}'$ over a $\mathbf{FO}$ signature $\mathbf{Sig}$ without function symbols, $\mathcal{I}$ is said to be a *subinterpretation* of $\mathcal{I}'$, in symbols $\mathcal{I} \Subset \mathcal{I}'$, whenever:

- $\mathbb{D}_{\mathcal{I}} \subseteq \mathbb{D}'_{\mathcal{I}}$,
- $S^{\mathcal{I}} = S^{\mathcal{I}'} \cap \mathbb{D}^n_{\mathcal{I}}$, for every n-ary relation symbol S, and
- $c^{\mathcal{I}} = c^{\mathcal{I}'}$, for every constant c.

[2] In general, closure under union of chains can be generalized to *formulas* by considering the stronger notion of *elementary subinterpretation*, but the current notion suffices for our proofs.

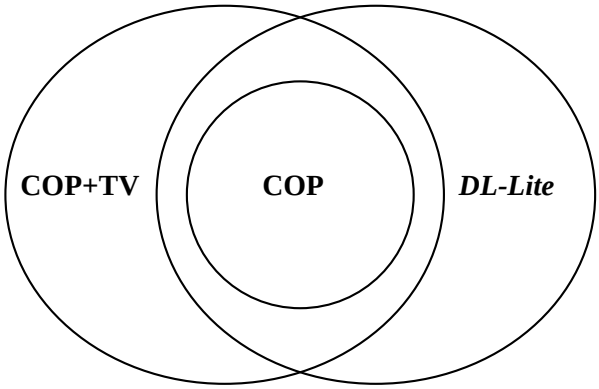


Figure 4.7: Relattive expressive power of the *DL-Lite* family, COP and COP+TV.

Definition 4.1.13 (Closure under unions of chains)**.** Let φ be a $\mathbf{FO}$ sentence. We say that φ is *closed under unions of chains* iff for any model $\mathcal{I}$, and any family $\{\mathcal{I}_i\}_{i\geq 0}$ of extensions of $\mathcal{I}$ s.t. $i \leq j$ implies $\mathcal{I}_i \trianglelefteq \mathcal{I}_j$, the structure $\mathcal{I}_\infty$, called *union interpretation* and defined below, is also a model of φ:

- $\mathbb{D}_{\mathcal{I}_\infty} := \bigcup_{i\geq 0} \mathbb{D}_{\mathcal{I}_i}$,
- $S^{\mathcal{I}_\infty} := \bigcup_{i\geq 0} S^{\mathcal{I}_i}$, and
- $c^{\mathcal{I}_\infty} := c^{\mathcal{I}_i}$, for $i \geq 0$.

We say that φ is a $\mathbf{FO}$ $\forall^*\exists^*$*-sentence* iff φ is of the form $\varphi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m \psi$, that is, a quantifier-free matrix ψ prefixed by n universal quantifiers followed by m existential quantifiers (with $n, m \geq 0$). Such $\forall^*\exists^*$-sentences are closed under unions of chains:

Theorem 4.1.14 ([CL03])**.** *A* $\mathbf{FO}$ *sentence φ is closed under unions of chains iff there exists an $\forall^*\exists^*$-sentence φ' logically equivalent to φ.*

Comparing the *DL-Lite* Family to the Fragments of English. In this section we prove that COP is strictly subsumed by *DL-Lite*$_{core}$ and hence by *DL-Lite*$_{\sqcap}$, *DL-Lite*$_F$, *DL-Lite*$_R$, *DL-Lite*$_{R,\sqcap}$ and *DL-Lite*$_{F,\sqcap}$. The other fragments overlap only in expressive power with *DL-Lite*$_{core}$. The general picture is summarized by Figure 4.7.

Theorem 4.1.15. *COP is contained in DL-Lite*$_{core}$*, and hence in every the logic of the DL-Lite family.*

Proof. Let $\Gamma \cup \Delta$ be a set of COP meaning representations, with Γ a set of quantified meaning representations and Δ a set of non-quantified meaning representations (a set of $\mathbf{FO}$ atomic sentences). We sketch a simulation mapping $\Gamma \cup \Delta$ to a *DL-Lite*$_{core}$ knowledge base $(\mathcal{O}_\Gamma, \mathcal{D}_\Delta)$ as follows.

We map (universal) sentences $\forall x(P(x) \Rightarrow \pm Q(x))$ to *DL-Lite*$_{core}$ inclusion assertions $A_P \sqsubseteq \pm A_Q$. By Skolemizing and dropping UNA over the new Skolem constants, we can map the meaning representations $\exists x(P(x) \wedge \pm Q(x)) \in \Gamma$ to the (database) assertions/facts $A_P(c')$ and $\pm A_Q(c')$, where c' is a (fresh) Skolem constant that does not occur in either Γ or Δ. Next, for each existing or newly introduced negative atom $\neg A_P(c)$, we *(i)* introduce a fresh unary predicate A'_P, *(ii)* a disjointness assertion $A'_P \sqsubseteq \neg A_P$ and *(iii)* map fact $\neg A_P(c)$ to fact $A'_P(c)$ and assertion $A'_P \sqsubseteq \neg A_P$. We get as a result the *DL-Lite*$_{core}$ knowledge base $(\mathcal{O}_\Gamma, \mathcal{D}_\Delta)$.

Clearly, for all interpretations $\mathcal{I}$,

$$\mathcal{I} \models \Gamma \cup \Delta \quad \text{iff} \quad \mathcal{I} \models (\mathcal{O}_\Gamma, \mathcal{D}_\Delta). \tag{†}$$

Furthermore, this simulation proceeds in time $\mathbf{O}(\#(\Gamma) + \#(\Delta))$, using $\mathbf{O}(\mathtt{log}\,\#(\Delta))$ space, where $\#(\Gamma)$ denotes the number of setences in Γ and $\#(\Delta)$ the number of constants occuring among the atomic sentences in Δ. □

Theorem 4.1.16. *DL-Lite$_{core}$, and hence every logic of the DL-Lite family, is not contained in COP.*

Proof. To show that this theorem holds, we exhibit a semantic property that is expressible in *DL-Lite$_{core}$* but not in COP. Consider the $\mathbf{FO}$ sentence

$$\varphi := \forall x(A(x) \Rightarrow \exists y r(x, y)),$$

which corresponds to *DL-Lite$_{core}$* assertions of the form $A \sqsubseteq \exists r$. The models of sentence φ are the interpretations $\mathcal{I}$ where every point in $A^\mathcal{I} \subseteq \mathbb{D}_\mathcal{I}$ is connected by $r^\mathcal{I} \subseteq \mathbb{D}_\mathcal{I} \times \mathbb{D}_\mathcal{I}$ to some (arbitrary) point in $\mathbb{D}_\mathcal{I}$, viz., the models where $A^\mathcal{I} \subseteq \{c \in \mathbb{D}_\mathcal{I} \mid \text{ exists } c' \in \mathbb{D}_\mathcal{I} \text{ s.t. } (c, c') \in r^\mathcal{I}\}$. But this semantic property cannot be expressed in COP because the signature $\mathbf{Sig}_{\text{COP}}$ of $\mathbf{L}_{\text{COP}}$ contains no relation symbols. □

Theorem 4.1.17. *COP+TV is not contained in either DL-Lite$_{R,\sqcap}$ or DL-Lite$_{F,\sqcap}$, and hence in no logic of the DL-Lite family.*

Proof. To prove this theorem, we show that *DL-Lite$_{R,\sqcap}$* and *DL-Lite$_{F,\sqcap}$* are closed under unions of chains, but not COP+TV. Assertions in the *DL-Lite* family yield $\mathbf{FO}$ $\forall^*\exists^*$-sentences. Hence, all these logics are closed under unions of chains.

Suppose by constradiction that COP+TV is contained in in either *DL-Lite$_{R,\sqcap}$* or *DL-Lite$_{F,\sqcap}$*. Then (modulo some translation/simulation $\cdot^t$), the same closure property should hold for COP+TV. In particular, the closure under unions of chains property should hold for the meaning representation

$$\exists x(P(x) \wedge \forall y(Q(y) \Rightarrow S(x, y)))$$

which, after prenexing, gives rise to the (equivalent) $\exists^*\forall^*$-sentence

$$\varphi := \exists x \forall y(P(x) \wedge (Q(y) \Rightarrow S(x, y))).$$

But this is impossible. To see this consider the following model $\mathcal{I}$ of φ:

- $\mathbb{D}_\mathcal{I} := \mathbb{N}$,
- $P^\mathcal{I} := Q^\mathcal{I} := \mathbb{D}_\mathcal{I}$, and
- $S^\mathcal{I} := \leq_\mathbb{N}$. (i.e. the usual loose order over positive integers).

Notice that $(\mathbb{N}, \leq_\mathbb{N})$ is well-founded and has 0 as least element; $\mathcal{I}$ is isomorphic to this structure. Define now a sequence $\{\mathcal{I}_i\}_{i\geq 0}$ of interpretations as follows:

- $\mathcal{I}_0$ is the model where
  - $\mathbb{D}_{\mathcal{I}_0} := \mathbb{D}_\mathcal{I} \cup \{c_0\}$,
  - $P^{\mathcal{I}_0} := Q^{\mathcal{I}_0} := \mathbb{D}_{\mathcal{I}_0}$, and
  - $S^{\mathcal{I}_0} := S^\mathcal{I} \cup \{(c_0, 0)\}$.
- $\mathcal{I}_{i+1}$ is the model where
  - $\mathbb{D}_{\mathcal{I}_{i+1}} := \mathbb{D}_{\mathcal{I}_i} \cup \{c_{i+1}\}$,

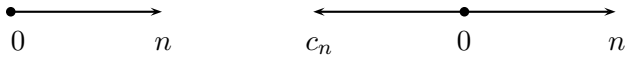


Figure 4.8: The interpretations $\mathcal{I}_0$ and $\mathcal{I}_\infty$.

- $P^{\mathcal{I}_{i+1}} := Q^{\mathcal{I}_{i+1}} := \mathbb{D}_{\mathcal{I}_{i+1}}$, and
- $S^{\mathcal{I}_{i+1}} := S^{\mathcal{I}_i} \cup \{(c_{i+1}, c_i)\}$.

Clearly, for all $i \geq 0$, $\mathcal{I}_i \subsetneq \mathcal{I}_{i+1}$. Hence $\{\mathcal{I}_i\}_{i\geq 0}$ is a chain ordered by $\subsetneq$. Finally, consider the union structure $\mathcal{I}_\infty$ for this chain. One can easily see that $\mathcal{I}_\infty$ is not a model of φ, since the relation $R^{\mathcal{I}_\infty}$ in $\mathcal{I}_\infty$ has no more a least element ($\mathcal{I}_\infty$ is now isomorphic to $(\mathbb{Z}, \leq_{\mathbb{Z}})$). □

Theorem 4.1.18. *DL-Lite$_{core}$, and hence every logic of the DL-Lite family, is not contained in COP+TV.*

Proof. In the *DL-Lite* family we can write *role-typing* assertions of the form $\exists r \sqsubseteq A$, or equivalently in $\mathbf{FO}$, sentence

$$\varphi := \forall x(\exists y r(x, y) \Rightarrow A(x)).$$

The models of φ belong to the class of interpretations $\mathcal{I}$ in which the domain of relation $r^{\mathcal{I}}$ is a subset of $A^{\mathcal{I}}$. On the other hand, COP+TV can only express finitely many semantic properties (i.e., classes of interpretations), since we can generate only finitely many $\mathbf{FO}$ sentences from its meaning representations (see [PHT06] and Table 2.1). By inspection one can see that none of such expressible properties coincides with the "role typing" property. Hence, role typing assertions are not expressible by COP+TV. □

Comparing the Controlled Language Constructs. Lite-English (with its two extensions Lite-English$_F$ and Lite-English$_R$) expresses the ontology languages *DL-Lite$_{core}$*, *DL-Lite$_\sqcap$*, *DL-Lite$_F$*, *DL-Lite$_R$*, *DL-Lite$_{F,\sqcap}$* and *DL-Lite$_{R,\sqcap}$*, thus inheriting their nice computational properties. These properties propagate to its function words:

- **(Quantification)** In Lite-English, universal quantification can occur only *once* and be followed by (possibly) $n \geq 0$ existential quantifiers. In COP+TV(+DTV) quantifiers may occur in any order. Furthermore it is restricted to subject $\mathbf{NP}$s
- **(Negation)** The Lite-English disallows negated facts. Negation is not Boolean: it can only occur on predicate $\mathbf{VP}$s and is expressed by the negative (left) determiner "no".
- **(Relatives)** Lite-English covers a restricted case of relative clauses, that neither COP nor COP+TV(+DTV) cover, which may occur only in subject $\mathbf{NP}$s.

Notice also that, by constrast to COP, COP+TV(+DTV), Lite-English can generate an infinite number of English utterances (it contains recursive constituents).

4.1.4 *DL-Lite$_\sqcap$* and Disjunction

A *closure* or *invariance property* is a relation $\sim_c$ over ($\mathbf{FO}$) interpretations, s.t., for each pair of interpretations $\mathcal{I}$ and $\mathcal{I}'$ and all $\mathbf{FO}$ formulas φ, if $\mathcal{I} \in$ *Mod*(φ) and $\mathcal{I} \sim_c \mathcal{I}'$, then $\mathcal{I}' \in$ *Mod*(φ). In such case, we say that φ is *closed under* $\sim_c$.

Closure properties can be used to characterize the *absolute* expressive power of a logic $\mathbf{L}$. We say that a logic $\mathbf{L}$ is *closed under* an invariance property $\sim_c$, whenever, for each $\mathbf{FO}$ formula φ, φ is (logically) equivalent to some $\varphi' \in \mathbf{L}$ iff φ is closed under $\sim_c$.

We can prove that no characterization theorem exists for assertions for *DL-Lite*$_\sqcap$ and a fortiori Lite-English. We rely on the fact that *DL-Lite*$_{R,\sqcap}$ cannot express disjunction:

Proposition 4.1.19. *Disjunction is not expressible in DL-Lite*$_{R,\sqcap}$.

Proof. *DL-Lite*$_{R,\sqcap}$ is contained in **HORN**, the set of **FO** Horn clauses [CGL$^+$07, CAKZ07], which cannot express disjunctions of the form $\varphi := A(c) \vee A'(c')$. Otherwise, let $\mathcal{H}$ with $\mathbb{D}_\mathcal{H} := \{c\}$ and $A^\mathcal{H} := \{c\}$ and $\mathcal{H}'$ with $\mathbb{D}_\mathcal{H} := \{c'\}$ and $A^\mathcal{H} := \{c'\}$ be two Herbrand models of φ. Clearly, $\mathcal{H}$ and $\mathcal{H}'$ are minimal (w.r.t. $\subseteq$) models of φ s.t. $\mathcal{H} \neq \mathcal{H}'$. But this is impossible, since **HORN** verifies the least (w.r.t. $\subseteq$) Herbrand model property ([Lal97], Proposition VI-2). □

Theorem 4.1.20. *There is no relation $\sim_c$ over interpretations such that, for every* **FO** *sentence φ, φ is equivalent to a DL-Lite*$_\sqcap$ *assertion iff φ is closed under $\sim_c$.*

Proof. Recall that a **FO** sentence is a **FO** formula with no free variables. Suppose the contrary and consider the sentence $A(c)$. Let $\mathcal{I}$ and $\mathcal{I}'$ be two structures s.t. $\mathcal{I} \sim_c \mathcal{I}'$ and suppose that $\mathcal{I} \models A(c)$. Clearly, this implies that $\mathcal{I} \models A(c) \vee A'(c)$. Since, on the other hand, by hypothesis, $A(c)$ is closed under $\sim_c$, it follows that $\mathcal{I}' \models A(c)$ too. But then, since $\mathcal{I}' \models A(c)$ implies $\mathcal{I}' \models A(c) \vee A'(c)$, this means that $A(c) \vee A'(c)$ is closed under $\sim_c$ and is a fortiori equivalent to some *DL-Lite*$_\sqcap$ assertion. But this is impossible, because disjunction is not expressible. □

4.2 GCQ-English

In this section we express *graph-shaped conjunctive queries*. Graph-shaped conjunctive queries are a slight generalization of TCQs which *(i)* allow for constants and *(ii)* allow for some simple loops on top of their tree structure. This is reflected in English by, on the one hand, proper names and, on the other hand, personal pronouns ("him", "himself", "it", "itself", "herself", if we consider gender).

Definition 4.2.1 (Graph-shaped conjunctive queries)**.** A *graph-shaped conjunctive query* is a CQ $\varphi(x)$ of distinguished variable x over a signature of relations of arity ≤ 2 where $\varphi(x)$ is inductively defined as follows.

$$\begin{aligned} \varphi(x) &\rightarrow A(x) \mid R(x,x) \mid R(x,c) \mid \exists y R(x,y) \mid \varphi'(x) \wedge \varphi''(x) \mid \exists y (R(x,y) \wedge \varphi'(y)) \\ R(x,y) &\rightarrow r(x,y) \mid r(y,x) \mid R(x,y) \wedge R'(x,y) \end{aligned} \tag{GCQ}$$

A *Boolean* GCQ is a query of the form $\exists x \varphi(x)$, where $\varphi(x)$ is as above.

Basically, a GCQ can be mapped into a labelled graph that is almost a tree, but where, in addition: *(i)* there can be loops over each node and *(ii)* there can be directed edges connecting nodes at level i, for $i \geq 2$, to its ancestor node at level $i - 2$. In general, variables and constants correspond to nodes and binary relations to directed edges.

GCQs are captured by the interrogative controlled language GCQ-English. Questions in GCQ-English fall under two main classes : *(i)* Wh-questions, that will map into non-Boolean GCQs and *(ii)* Y/N-questions, that will map into Boolean GCQs. Figure 4.9 shows GCQ-English's grammar. Some basic morpho-syntactic and semantic features are attached to (some) constituents. The feature $\cdot^-$ means that the constituents is of negative polarity. Absence of features indicates that constituents are in positive polarity. Notice that as for Lite-English, we disregard all other morphosyntactic features.

Personal pronouns ("him") co-refer with the closest **NP** in *argument* position. Reflexive pronouns ("himself), like relative pronouns, co-refer with their closest **NP** in *subject* position. The

(Phrase structure rules)	(Semantic actions)
$\mathbf{Q}_{wh} \to \mathbf{Intpro}\ \mathbf{N}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \tau(\mathbf{Intpro})(\tau(\mathbf{N}_i))(\tau(\mathbf{S}_{g_i}))$
$\mathbf{Q}_{wh} \to \mathbf{Intpro}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \tau(\mathbf{Intpro}_i)(\tau(\mathbf{S}_{g_i}?))$
$\mathbf{Q}_{Y/N} \to$ does $\mathbf{NP}_i^-\ \mathbf{VP}_i^-?$	$\tau(\mathbf{Q}_{Y/N}) := \tau(\mathbf{NP}_i^-)(\tau(\mathbf{VP}_i^-))$
$\mathbf{Q}_{Y/N} \to$ is $\mathbf{NP}_i\ \mathbf{VP}_i$	$\tau(\mathbf{Q}_{Y/N}) := \tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i))$
$\mathbf{S}_{g_i} \to \mathbf{NP}_{g_i}\mathbf{VP}_i$	$\tau(\mathbf{S}_{g_i}) := \tau(\mathbf{NP}_{g_i})(\tau(\mathbf{VP}_i))$
$\mathbf{N}_i \to \mathbf{Adj}\ \mathbf{N}_i$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_i))$
$\mathbf{N}_i \to \mathbf{N}_i\ \mathbf{Relp}_i\ \mathbf{S}_{g_i}$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Relp}_i)(\tau(\mathbf{N}_i))(\tau(\mathbf{S}_{g_i}))$
$\mathbf{VP}_i \to$ is $\mathbf{Adj}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{Adj}_i)$
$\mathbf{VP}_i \to$ is a $\mathbf{N}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{N}_i)$
$\mathbf{VP}_i \to \mathbf{VP}_i\ \mathbf{Crd}\ \mathbf{VP}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}_i))(\tau(\mathbf{VP}_i))$
$\mathbf{VP}_i^- \to \mathbf{VP}_i^-\ \mathbf{Crd}\ \mathbf{VP}_i^-$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}_i^-))(\tau(\mathbf{VP}_i^-))$
$\mathbf{VP}_i \to \mathbf{IV}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{IV}_i)$
$\mathbf{VP}_i^- \to \mathbf{IV}_i^-$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{IV}_i^-)$
$\mathbf{VP}_i \to \mathbf{TV}_{i,i+1}\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{TV}_{i,i+1})(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{VP}_i^- \to \mathbf{TV}_{i,i+1}^-\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{VP}_i^-) := \tau(\mathbf{TV}_{i,i+1}^-)(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{NP}_i \to \mathbf{Pro}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Pro}_i)$
$\mathbf{NP}_i^- \to \mathbf{Pro}_i^-$	$\tau(\mathbf{NP}_i^-) := \tau(\mathbf{Pro}_i^-)$
$\mathbf{NP}_i \to \mathbf{Det}\ \mathbf{N}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Det})(\tau(\mathbf{N}_i))$
$\mathbf{NP}_i \to \mathbf{Pn}_i$	$\tau(\mathbf{NP}_i) :=: \tau(\mathbf{Pn}_i)$
$\mathbf{NP}_{g_i} \to t_i$	$\tau(\mathbf{NP}_{g_i}) := \lambda P.P(x)$

(Function lexicon)

$\mathbf{Det} \to$ some	$\tau(\mathbf{Det}) := \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))$
$\mathbf{Pro}_i \to$ somebody	$\tau(\mathbf{Pro}_i) := \lambda P.\exists x P(x)$
$\mathbf{Pro}_i^- \to$ anybody	$\tau(\mathbf{Pro}_i^-) := \lambda P.\exists x.P(x)$
$\mathbf{Crd} \to$ and	$\tau(\mathbf{Crd}) := \lambda P.\lambda Q.\lambda x.(P(x) \wedge Q(x))$
$\mathbf{Relp}_i \to$ who	$\tau(\mathbf{Relp}_i) := \lambda P.\lambda x.P(x)$
$\mathbf{Intpro} \to$ which	$\tau(\mathbf{Intpro}) := \lambda P.\lambda Q.\lambda x.P(x) \wedge Q(x)$
$\mathbf{Intpro}_i \to$ who	$\tau(\mathbf{Intpro}_i) := \lambda P.\lambda x.P(x)$

(Content lexicon)

$\mathbf{Pro}_{i-2} \to$ him	$\tau(\mathbf{Pro}_{i-2}) := \lambda P.P(x)$
$\mathbf{Pro}_{i-1} \to$ himself	$\tau(\mathbf{Pro}_{i-1}) := \lambda P.P(x)$
$\vdots$	$\vdots$

Figure 4.9: GCQ-English.

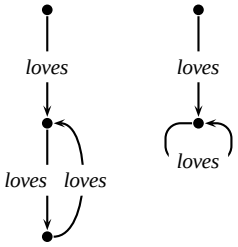


Figure 4.10: Left: A GCQ for "Does somebody love somebody who loves him?". **Right:** A GCQ for "Does somebody love somebody who loves himself?"

co-reference of pronouns is captured by playing with the indexes of constituents. A reflexive pronoun (resp. a personal pronoun) dominated by an $\mathbf{NP}_i$ of index i co-refers with an $\mathbf{NP}_{i-1}$ of index $i-1$ (resp. an $\mathbf{NP}_{i-2}$ of index $i-2$). The grammar of GCQ-English assigns consecutive integers as indexes to denoting constituents. The definition of grammar derivations can be easily extended to cover this phenomenon, by "unifying" constituents with the same indexes. For simplicity, we only specify partially the (arbitrarily large) content lexicon.

Example 4.2.2. A typical Boolean GCQ over, say, the constant Mary and the binary predicates *loves* and *hates* is

$$\exists x(\textit{loves}(\text{Mary}, x) \wedge \exists y \textit{hates}(x, y)) \tag{4.4}$$

which we express with the controlled language Y/N-question

Does Mary love somebody who hates somebody? (4.5)

A typical non-Boolean graph-shaped query over the same set of relational symbols is

$$\exists y(\textit{loves}(x, y) \wedge \textit{hates}(y, x)) \tag{4.6}$$

which we express with the controlled language Wh-question (containing an anaphoric pronoun)

who loves somebody who hates him? (4.7)

On the other hand

*Which teacher gives a lesson to his pupils? (4.8)

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

Similarly, the Y/N questions

Does somebody love somebody who loves him? (4.9)

and

Does somebody love somebody who loves himself? (4.10)

are GCQ-English questions. The personal pronoun "him" co-refers with the first "somebody", which means that question (4.9) translates into

$$\exists x(\exists y(\textit{loves}(x, y) \wedge \textit{loves}(y, x))) \tag{4.11}$$

whereas question (4.10) translates into

$$\exists x(\exists y(\textit{loves}(x, y) \wedge \textit{loves}(y, y))). \tag{4.12}$$

Figure 4.10 shows how these co-references are reflected by the graph-structure of the query expressed. ♣

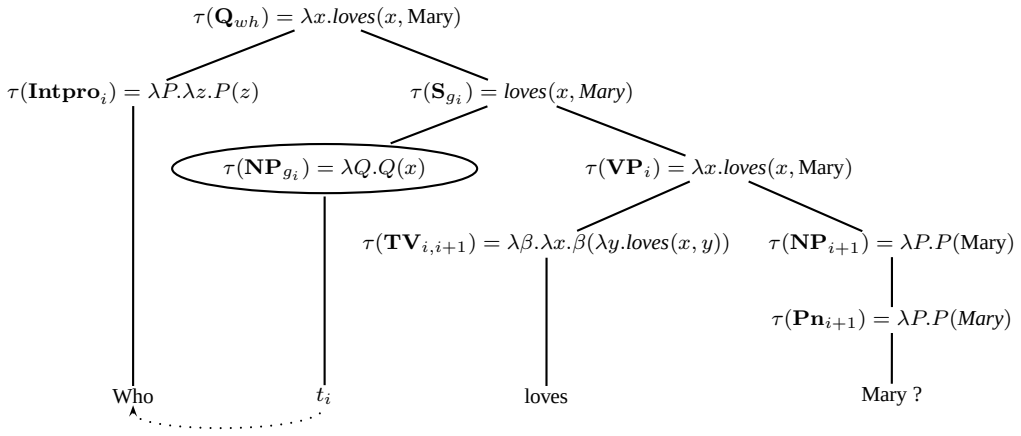


Figure 4.11: Translating "Who loves Mary?".

Theorem 4.2.3 (Expressing GCQs)**.** *For every question Q in GCQ-English, there exists a GCQ φ s.t. $\tau(Q) \equiv_s \varphi$. Conversely, every GCQ φ is the image by $\tau(\cdot)$ of some question Q in GCQ-English.*

Proof. ($\Rightarrow$) As for Lemma 4.1.2, we need to show, by mutual induction on the length n of derivations rooted in $\mathbf{N}$s and $\mathbf{VP}$s, that

$$\begin{array}{c}\text{for every } \mathbf{N} \text{ (resp. } \mathbf{VP}\text{) in GCQ-English there}\\ \text{exists a GCQ } \varphi(x) \text{ s.t. } \tau(\mathbf{VP}) \equiv_s \varphi(x). \text{ (resp. } \tau(\mathbf{N}) \equiv_s \varphi(x)).\end{array} \quad (\dagger)$$

The basis (i.e., $n = 0$) is trivial. For the inductive step (i.e., $n = k + 1$) we will show only one case. All the other cases are analogous. Let $\mathbf{VP}_i \Longrightarrow \mathbf{TV}_{i,j}\,\mathbf{NP}_j \Longrightarrow \mathbf{TV}_{i,j}\,\mathbf{Det}\,\mathbf{N}_j \Longrightarrow^{k-1} R$s some w, with $\mathbf{N}_i \Longrightarrow^{k-2} w$. By IH, $\tau(\mathbf{N}_i) \equiv_s \varphi(x)$. Therefore,

$$\begin{array}{rl}\tau(\mathbf{VP}_i) =_{df} & \lambda\beta.\lambda x.\beta(\lambda y.r(x,y))(\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z)))\\ & (\tau(\mathbf{N}_i))\\ =_{ih} & \lambda\beta.\lambda x.\beta(\lambda y.r(x,y))(\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z)))\\ & (\lambda w.\varphi(w))\\ \rhd & \lambda x.\exists y(r(x,y) \wedge \varphi(y))\end{array}$$

which is a formula 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 will only deal with one of the four possible cases. Let $\mathbf{Q}_{wh} \Longrightarrow \mathbf{Intpro}\,\mathbf{N}_i\,\mathbf{S}_{g_i} \Longrightarrow \mathbf{Intpro}\,\mathbf{N}_i\,\mathbf{VP}_i \Longrightarrow^*$ which $w'\,w''$. Then

$$\begin{array}{rl}\tau(\mathbf{Q}_{wh}) =_{df} & \lambda P.\lambda Q.\exists x(P(x) \wedge Q(x))(\lambda z.\varphi'(z))(\lambda w.\varphi''(w))\\ \rhd & \lambda x.\varphi'(x) \wedge \varphi''(x)\end{array}$$

which is structurally equivalent to a GCQ. Note that, as before, we discard, when parsing, all possible parse states where constituents do not satisfy co-indexing, polarity and typing constraints (i.e., when features do not unify).

($\Leftarrow$) The proof is analogous to that of Lemma 4.1.3: we prove, by induction on $\varphi(x)$ (a non-Boolean GCQ φ of distinguished variable x) that

$$\begin{array}{c}\text{for each GCQ } \varphi(x) \text{ we can construct a GCQ-English}\\ \text{constituent } w \text{ s.t. } \varphi(x) \text{ is the image of } w \text{ by } \tau(\cdot)\end{array} \quad (\ddagger)$$

up to structural equivalence. Claim (‡) entails that "which w?" and/or "who w?" is the question we are looking for. Recall that unary predicates A are captured by $\mathbf{N}$s of meaning representation

$$\lambda x.A(x),$$

relation symbols r by $\mathbf{TV}$s of meaning representation

$$\lambda\beta.\lambda x.\beta(\lambda y.r(x,y))$$

and individual constants c by $\mathbf{Pn}$s of meaning representation

$$\lambda P.P(c).$$

Similarly, relative pronouns ($\mathbf{Relp}$s like "who") and conjunctions ($\mathbf{Crd}$s like "and"), of meaning representation

$$\lambda P.\lambda Q.\lambda x(P(x) \wedge Q(x)),$$

express conjunctions, whereas the $\mathbf{Pro}$ "somebody", of meaning representation

$$\lambda P.\exists x P(x)$$

expresses existential quantification.

By combining together such meaning representations following $\mathbf{Ho}$ typing rules as we did in Lemma 4.1.3, it is easy to see that:

- **(Basis)** $\varphi(x)$ is of the form $A(x)$, $r(x,x)$, $r(x,c)$ or $\exists y r(x,y)$. Accordingly, it is the image of either "is an A", "Rs himself", "rs c" or "rs somebody".
- **(Inductive step)** We will only look at one case. Let $\varphi(x)$ be of the form $\exists y(r(x,y) \wedge \varphi'(y))$. Assume in addition that $R(x,y) := r_1(x,y) \wedge \cdots \wedge r_k(x,y)$ Then $\varphi(x)$ is the image of "r_1s somebody who w and r_2s him and ... and r_ks him", by IH on $\varphi'(y)$, up to structural equivalence. The other cases are similar.

This closes the proof. □

4.3 Data and Combined Complexity

In this section we study briefly the computational complexity of KBQA for Lite-English and GCQ-English. Since Lite-English and GCQ-English express, when taken together, a restricted case of KBQA for *DL-Lite*$_{R,\sqcap}$ ontologies and UCQs ((U)GCQs and (U)TCQs are a fragment of (U)CQs), they inherit their computational properties. This observation holds both for the data and the combined complexity of KBQA.

Theorem 4.3.1. KBQA *is in* **LSpace** *w.r.t. data complexity for Lite-English and GCQ-English (in fact, for UCQs).*

Proof. Let $\mathcal{S}$ and $\mathcal{F}$ be a set of Lite-English declarations and facts, respectively. We know by Theorem 4.3.1 that KBQA is in **LSpace** in data complexity when we consider UCQs and *DL-Lite*$_{\sqcap}$ knowledge bases. On the other hand, compositional translations $\tau(\cdot)$ encode $\mathcal{F}$ into a content lexicon using space logarithmic in the number $\#(\mathcal{F})$ of proper nouns (i.e., object names, to which a lexical entry is associated). Since we have shown that Lite-English expresses *DL-Lite*$_{\sqcap}$ (Theorem 4.1.4) and GCQ-English GCQs (Theorem 4.2), the result follows. □

Theorem 4.3.2. KBQA *is in* **LSpace** *w.r.t. data complexity for COP and GCQ-English (in fact, for UCQs).*

Proof. Since COP is contained in *DL-Lite* by Theorem 4.1.15, membership in **LSpace** for KBQA is a corollary of Theorem 4.3.1. □

Furthermore, by applying Theorem 3.3.6, we get, as an immediate corollary of these proofs:

Corollary 4.3.3. KBSAT *is in* **LSpace** *in data complexity for COP and Lite-English knowledge bases.*

A *perfect reformulation* is an algorithm that takes as input a description logic ontology $\mathcal{O}$ and a UCQ φ of arity n and rewrites φ w.r.t. $\mathcal{O}$ into a UCQ $\varphi_{\mathcal{O}}$ s.t., for every database $\mathcal{D}$ and every sequence $\bar{c}$ of n constants it holds that: $(\mathcal{O}, \mathcal{D}) \models \varphi(\bar{c})$ iff $\mathcal{I}(\mathcal{D}) \models \varphi_{\mathcal{O}}(\bar{c})$.

Proposition 4.3.4 ([CdV$^+$06])**.** *A perfect reformulation exists for DL-Lite*$_{R,\sqcap}$.

Theorem 4.3.5. KBQA *for empty ontologies is* **NPTime***-hard in combined complexity. It is in* **NPTime** *for DL-Lite*$_{R,\sqcap}$.

Proof. (**Membership**) Let $(\mathcal{O}, \mathcal{D})$ be a *DL-Lite*$_{R,\sqcap}$, knowledge base, φ a UCQ of arity n and $\bar{c}$ a sequence of n constants. Let $\varphi(\bar{c})$ denote the grounding of φ by c and suppose that $(\mathcal{O}, \mathcal{D}) \models \varphi(\bar{c})$.

We know that $\mathcal{O}$ can be "compiled" into φ by a perfect reformulation, yielding a UCQ $\varphi_{\mathcal{O}}(\bar{c}) := \varphi_1^{\mathcal{O}}(\bar{c}, \bar{y}_1) \vee \cdots \vee \varphi_k^{\mathcal{O}}(\bar{c}, \bar{y}_k)$. Guess in time polynomial in $\#(\mathcal{D})$, $\#(\mathcal{O})$ and $|\varphi|$, where $|\varphi|$ denotes the number of symbols in UCQ φ, a disjunct $\varphi_i^{\mathcal{O}}(\bar{c}, \bar{y}_i)$, for some $i \in [1, k]$. Clearly, $(\mathcal{O}, \mathcal{D}) \models \varphi(\bar{c})$ iff $\mathcal{I}(\mathcal{D}), \gamma \models \varphi_i^{\mathcal{O}}(\bar{c}, \bar{y}_i)$, for some assignment γ. Guess now an assignment $\gamma \colon \mathit{FV}(\varphi_i^{\mathcal{O}}) \to \mathbb{D}_{\mathcal{I}(\mathcal{D})}$. This can be done in time polynomial in $|\varphi|$, $\#(\mathcal{O})$ and $\#(\mathcal{D})$. Finally, check in time polynomial on $\#(\mathcal{D})$, $\#(\mathcal{O})$ and $|\varphi|$ whether $\mathcal{I}(\mathcal{D}), \gamma \models \varphi_i^{\mathcal{O}}(\bar{c}, \bar{y}_i)$.

(**Hardness**) By reduction from the graph homomorphism problem, where, given two graphs $\mathcal{G} = (\mathbb{V}, \mathbb{E})$ and $\mathcal{G}' = (\mathbb{V}', \mathbb{E}')$ we ask whether there exists an homomorphism h from $\mathcal{G}$ to $\mathcal{G}'$. A graph homomorphism, we recall, is a function $h \colon \mathbb{V} \to \mathbb{V}'$ s.t. for all $(u, v) \in \mathbb{E}$, $(h(u), h(v)) \in \mathbb{E}'$. This problem is known to the **NPTime**-complete [GJ79]. We will consider empty ontologies. Polynomially encode $\mathcal{G}$ and $\mathcal{G}'$ as follows:

- for each $(u, v) \in \mathbb{E}$, add the fact $e(c_u, c_v)$ to the database $\mathcal{D}_{\mathcal{G}}$,
- for each $(u', v') \in \mathbb{E}'$, add the ground atom $e'(c_{u'}, c_{v'})$ to the Boolean UCQ $\varphi_{\mathcal{G}'}$.

We now claim that

$$\text{there exists } h \text{ from } \mathcal{G}' \text{ to } \mathcal{G} \quad \text{iff} \quad () \in \mathit{cert}(\varphi_{\mathcal{G}'}, \emptyset, \mathcal{D}_{\mathcal{G}}). \tag{$\dagger$}$$

Consider now an *empty* perfect reformulation for *DL-Lite*$_{R,\sqcap}$. It follows that $(\emptyset, \mathcal{D}_{\mathcal{G}}) \models \varphi_{\mathcal{G}'}$ iff $\mathcal{I}(\mathcal{D}_{\mathcal{G}}) \models \varphi_{\mathcal{G}'}$ iff $\mathcal{I}(\mathcal{D}_{\mathcal{G}}) \models \varphi_{\mathcal{G}'}$ for some assignment γ. Now, clearly, $\mathcal{I}(\mathcal{D}_{\mathcal{G}}) = \mathcal{G}$. Thus, the composition $\cdot^{\mathcal{I}(\mathcal{D}_{\mathcal{G}})} \circ \gamma$ can be seen as an homomorphism mapping $\varphi_{\mathcal{G}'}$ to $\mathcal{G}$. Finally, given that $\varphi_{\mathcal{G}'}$ encodes $\mathcal{G}'$, the claim follows. □

It immediately follows:

Corollary 4.3.6. KBQA *is* **NPTime***-complete in combined complexity for Lite-English and COP knowledge bases and GCQ-English questions (in fact, for UCQs).*

4.4 Summary

In this chapter we have defined the declarative controlled languages Lite-English, Lite-English$_F$ and Lite-English$_R$, expressing the *DL-Lite* family of description logics, and an interrogative controlled language, GCQ-English, expressing GCQs and TCQs. This can be achieved using standard **Ho** meaning representations, by constraining the behavior of syntactic constituents.

In the case of the *DL-Lite* family, constraining the behavior of syntactic constituents involves the widespread use of subcategorization. Constituents have to be subcategorized into left and right constituents depending on whether they are meant to express a right C_r or a left C_l concept in an concept inclusion assertion $C_l \sqsubseteq C_r$. We recall that in the *DL-Lite* family left and right concepts are given a separate syntax. This distinction is essential for the good computational properties of these description logics.

We have have studied the relative expressive power of Lite-English and of the *DL-Lite* family by comparing it with the fragments of English. We have shown that *(i)* Lite-English contains COP and that *(ii)* Lite-English overlaps with COP+TV. We have also shown that absolute expressive power of *DL-Lite*$_\sqcap$ ontologies and assertions cannot be, however, characterized, and that, as a result, neither can the absolute expressive power of Lite-English.

Last, but not least, we have studied the data and combined complexity of KBSAT and of KBQA w.r.t. GCQs and Lite-English.

Chapter 5

Expressing Aggregate Queries

In earlier chapters we have shown how to express TCQs in controlled language, covering a significant fragment of UCQs, which make up about 80% of database queries [AHV95, EN04]. Furthermore, we noted in Chapter 3 (recall Theorems 3.3.5 and 3.3.4) that such queries can be processed efficiently by both database engines and OBDASs [CdV$^+$06] based on the *DL-Lite* family of description logics. Controlled languages such as GCQ-English show that TCQs can be expressed quite naturally with English Wh- and Y/N-questions allowing for *(i)* existential generalized determiners, *(ii)* arbitrary nesting of subordinated (with gaps being filled by its closest head **NP**) clauses and *(iii)* **VP** and **N** coordination (conjunction). However, over some domains of interest, users might be interested in issuing more complex information requests. Consider the student domain one more time. Recall Example 3.2.4 from Chapter 3. Suppose we have now a richer ontology $\mathcal{O}_s$ of the domain as shown Figure 5.1. Suppose, in addition, that now $\mathcal{D}_s$ contains the following tables

takesCourse

SName	Course
Luca	TOC
Luca	ADS
James	German

Student

SName
Luca
James

comesFrom

SName	CoName
Luca	Italy
James	UK

Course

CName	Cred
TOC	4
ADS	4
German	0

Country

CoName
Italy
UK

where "TOC" stands for Theory of Computing and "ADS" for Algorithms and Data Structures. A user might want to mine this information and extract some very basic statistics, e.g., count how many of the enrolled students attended lectures. Suppose, finally, that she intends to do this through a controlled English interface. She would ask to the OBDAS $(\mathcal{O}_s, \mathcal{D}_s)$ the question Q_s

Which is the number of distinct students per country who study some course? (5.1)

which the system would translate into an *aggregate* SQL query

```
SELECT cf.CoName,COUNT(DISTINCT(cf.SName))
FROM comesFrom cf, attends at
WHERE cf.SName=at.SName)
GROUP BY cf.CoName
```
(5.2)

to be evaluated over $(\mathcal{O}_s, \mathcal{D}_s)$. Aggregates naturally arise in domains and systems containing numerical data, e.g., geographical information systems, or whenever we want to mine a statistic of any kind from a dataset ([AHV95], Chapter 7). SQL aggregate queries extend the syntax of `SELECT-PROJECT-JOIN-UNION` queries with *aggregation functions* (`SUM`, `MAX`, `COUNT`, etc), `GROUP BY` and `HAVING` clauses. This raises three problems:

1. We need to know which is the semantics of aggregate queries in OBDASs.

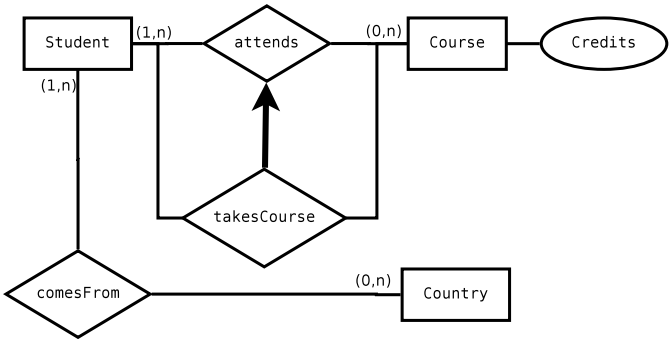


Figure 5.1: A conceptual diagram $\mathcal{O}_s$ of the student domain.

2. We need to know how to express such aggregate queries in controlled English.
3. We need to know whether the queries and the controlled English questions scale to data.

The semantics of aggregate queries in relational databases is well-understood, less is known instead about their semantics in OBDASs. Relational engines evaluate aggregate queries by *(i)* grouping database values into bags known as *groups* (defined by the `GROUP BY` clause and the `HAVING` clause that acts as a filter over the group(s)) and *(ii)* applying (bag-valued) aggregation functions over those groups, and *(iii)* returning the answers obtained (see [CNS07] and [AHV95], Chapter 7). A naive solution would be to reduce the semantics of aggregate queries over OBDASs to the database case by analogy to UCQs. Indeed, Proposition 3.2.5 from Chapter 3 says that the certain answers of a UCQ over an OBDAS are exactly those tuples that are answers over every database compatible with the OBDAS. But if we apply this definition, the answers of query (5.2) will always be empty. This is because an aggregate query can give rise to a different group (and a different aggregate value) over each compatible database [CNKT08]. A detailed example of this is given later in Example 5.1.6.

We do not know much either on how to express aggregate queries in natural language. A reasonable assumption is that aggregation functions are conveyed by English *definite* **NP**s, such as "the highest **N**", "the total number of **N**s", "the average number of **N**", etc. The Geoquery corpus[1] is a corpus of English US geographical questions (e.g., "Which is the longest river in Iowa?"): by assuming that questions containing "some", relatives, "and" plus possibly "or" as function words express UCQs, corpus statistics suggest that *(i)* aggregate determiners occur frequently, *(ii)* they occur in combination with UCQ constructs and *(iii)* they occur way more frequently than questions with negations like "is not" or "does not" (see Table 5.1). However, it is not immediate to formalize these intuitions in formal semantics.

It is, however, clear, that no answer to the third problem can be found without answering the first two. That is, without proposing a reasonable semantics for the queries and for the natural language questions.

To tackle these three problems, we will pursue in this chapter the following strategy. We will define a class of aggregate queries, *aggregate tree-shaped queries*, that extend the syntax of TCQs with SQL aggregation functions. Such queries provide a logic-based declarative specification of a significant class of SQL aggregate queries. Next, we will proceed to extend (and modify) the traditional certain answers semantics of UCQs over OBDASs to aggregate tree-shaped queries[2]. Thirdly, we will define a controlled fragment of aggregates questions that expresses precisely this

[1] `http://www.cs.utexas.edu/~ml/geo.html`

[2] Such syntax and semantics is derived from joint work with D. Calvanese, E. Kharlamov and W. Nutt on so-called *epistemic aggregate queries* in [CNKT08].

	UCQs	UCQs + Agg	UCQs + Neg
Questions	34.54%	65.35%	0.11%

Table 5.1: Frequency of questions "expressing" aggregate queries in the Geoquery corpus.

class of queries. Finally, modulo compositionality, we will study data complexity.

Note that we consider an ontology $\mathcal{O}$ to be an arbitrary set of $\mathbf{FO}$ axioms, written in one of many possible syntax such as ER diagrams, description logic ontologies or controlled English. We proceed analogously for databases $\mathcal{D}$, which we will write down sometimes in description logic format as sets of ground facts (or $\mathbf{FO}$ atomic sentences) or as relational database instances. We extend the notion of a knowledge base $(\mathcal{O}, \mathcal{D})$ accordingly. We enforce, however, the semantic assumptions of ontology languages, UNA, SDA and SNA, introduced in Chapter 4. Part of the results of this chapter were first published in [TC09] and [Tho08].

5.1 Aggregate Tree-Shaped Queries

Aggregate tree-shaped queries provide a $\mathbf{FO}$-based declarative specification of a significant fragment of SQL aggregate queries as studied in [CNKT08, CNS07, CNS03]. In particular, their their semantics is an instantiation of the epistemic certain answers semantics for epistemic aggregate queries defined and studied in [CNKT08].

5.1.1 Syntax

We consider now the following standard SQL *aggregation functions*, viz., $\mathbf{max}$, $\mathbf{min}$, $\mathbf{count}$, $\mathbf{cntd}$, $\mathbf{sum}$ and $\mathbf{avg}$. In what follows $\mathbf{agg}$ will denote an arbitrary aggregation function. Given this, we call an *aggregation term* any expression of the form $\mathbf{agg}(y)$, where y is called an *aggregation variable*.

Definition 5.1.1 (Aggregate tree-shaped query)**.** An *aggregate tree-shaped query* is an expression of the form

$$\varphi := \{(x, \mathbf{agg}(y)) \mid \psi\} \tag{ATCQ}$$

where x is called a *grouping variable*, $\mathbf{agg}(y)$ is an aggregation term, and ψ is the the query's *body*, which is a formula

$$\psi := \psi_1(x) \wedge r(x, y) \wedge \psi_2(y)$$

with ψ_1 a TCQ rooted in x, ψ_2 a TCQ rooted in y, $r(x, y)$ an atom, $\{x, y\} = \mathit{FV}(\psi)$ and $y \neq x$. The *core* $\tilde{\varphi}$ of φ is defined as the *quantifier-free* version of its body ψ (i.e., a quantifier-free CQ).

Example 5.1.2. Consider the $\mathbf{cntd}$ ATCQ

$$\begin{aligned}\varphi_s := \{(x, \mathbf{cntd}(y)) \mid \exists z(&\mathit{Student}(y) \wedge \mathit{attends}(y, z) \wedge \mathit{Course}(z) \wedge \\ &\wedge \mathit{comesFrom}(y, x) \wedge \mathit{Country}(x))\}.\end{aligned} \tag{5.3}$$

The ATCQ φ_s captures the SQL query (5.2) from the preamble of this chapter. The CQ

$$\tilde{\varphi}_s := \mathit{Student}(y) \wedge \mathit{attends}(y, z) \wedge \mathit{Course}(z) \wedge \mathit{comesFrom}(y, x) \wedge \mathit{Country}(x) \tag{5.4}$$

is its core. ♣

5.1.2 Certain Answers Semantics

In this section we extend the semantics of SQL aggregation functions to ontologies and knowledge bases. In relational databases SQL aggregation functions are computed over an specific kind of bag, known as a group, and return a numeric value. Bags (or multisets) are a generalization of the notion of set in which the same element may occur repeatedly many times.

Definition 5.1.3 (Bags)**.** Let X be a set. A *bag* or *multiset* B (over X) is a function $B\colon X \to \mathbb{N} \cup \{\infty\}$. The integer $B(x)$ is called the *multiplicity* of x in B. The *additive union* $B \uplus B'$ of two bags over X is the bag where $(B \uplus B)(x) := B(x) + B'(x)$, for all $x \in X$. The *carrier* $\xi(B)$ of a bag B over X is the set $\{x \in X \mid B(x) \geq 1\}$. Bags can be denoted by extension or intension using the special brackets $\{\!|\cdot|\!\}$.

Definition 5.1.4 (Aggregation functions)**.** Let $\mathbf{B}[X]$ denote the set of all bags over set X. An aggregation function is one of the following functions:

$$\mathbf{max}\colon \mathbf{B}[X] \to \mathbb{Q} \cup \{\infty\} \text{ s.t.}$$
$$\mathbf{max}(B) := \begin{cases} \max\limits_{x \in \xi(B)} x, \text{ if it exists,} \\ \infty, \text{ otherwise.} \end{cases}$$

$$\mathbf{min}\colon \mathbf{B}[X] \to \mathbb{Q} \cup \{\infty\} \text{ s.t.}$$
$$\mathbf{min}(B) := \begin{cases} \min\limits_{x \in \xi(B)} x, \text{ if it exists,} \\ \infty, \text{ otherwise.} \end{cases}$$

$$\mathbf{sum}\colon \mathbf{B}[X] \to \mathbb{Q} \cup \{\infty\} \text{ s.t.}$$
$$\mathbf{sum}(B) := \begin{cases} \sum\limits_{x \in \xi(B)} B(x) \cdot x, \text{ if defined and } B \text{ is finite,} \\ \infty, \text{ otherwise.} \end{cases}$$

$$\mathbf{count}\colon \mathbf{B}[X] \to \mathbb{N} \cup \{\infty\} \text{ s.t.}$$
$$\mathbf{count}(B) := \begin{cases} \sum\limits_{x \in \xi(B)} B(x), \text{ if defined and } B \text{ is finite,} \\ \infty, \text{ otherwise.} \end{cases}$$

$$\mathbf{avg}\colon \mathbf{B}[X] \to \mathbb{Q} \cup \{\infty\} \text{ s.t.}$$
$$\mathbf{avg}(B) := \begin{cases} \frac{\mathbf{count}(B)}{\#(X)}, \text{ if defined and } B \text{ is finite,} \\ \infty, \text{ otherwise.} \end{cases}$$

Notice that ∞ is a special value that stands for the cases in which the value of the aggregate function is not an integer or a rational number or is undefined. The *multiplicity insensitive* siblings $\mathbf{avgd}$, $\mathbf{mind}$, $\mathbf{maxd}$ $\mathbf{sumd}$ and $\mathbf{cntd}$ of aggregation functions are defined by composing aggregation functions and the carrier operation, e.g., $\mathbf{cntd}(B) := \mathbf{count}(\xi(B))$.

Groups intuitively collect the values of a (numerical or symbolic) attribute w.r.t. some given object in a database, which acts as the group identifier (see [EN04], Chapter 8). The syntax and semantics of ATCQs respect these features. In ATCQs, grouping variables stand for (or are bound to) such SQL group identifiers. On the other hand, aggregate variables stand for the attributes upon which groups are defined [CNS03, CNS07].

Definition 5.1.5 (Groups and database answers)**.** Let φ be an ATCQ of grouping variable x and aggregation variable y. Let $\mathcal{D}$ be a database. The *group* of tuple c is the bag

$$G_c := \{\!|\gamma(y) \mid c = \gamma(x), \gamma \in \mathit{Sat}_{\mathcal{D}}(\tilde{\varphi})|\!\}. \qquad \textbf{(Group)}$$

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

$$\mathit{ans}_a(\varphi, \mathcal{D}) := \{(\gamma(x), \mathbf{agg}(G_{\gamma(x)})) \mid \gamma \in \mathit{Sat}_{\mathcal{D}}(\tilde{\varphi})\}. \qquad \textbf{(Aans)}$$

However, contrary to databases, knowledge bases deal with incomplete information and work under the open world assumption (OWA), i.e., their databases are a partial description of the domain of interest that the ontology "completes" by intuitively characterizing the space of all the databases compatible with it [CNKT08]. In each such database an object may possess different attributes, giving rise to different groups and different values for aggregation functions. This precludes our naively applying the certain answers semantics for UCQs.

Example 5.1.6. We can encode the student $\mathcal{D}_s$ relational database from the preamble into the set of facts

$\mathit{takesCourse}(\text{Luca}, \text{TOC})$ $\quad$ $\mathit{hasCredits}(\text{TOC}, 4)$ $\quad$ $\mathit{comesFrom}(\text{Luca}, \text{Italy})$
$\mathit{takesCourse}(\text{Luca}, \text{ADS})$ $\quad$ $\mathit{hasCredits}(\text{ADS}, 4)$ $\quad$ $\mathit{comesFrom}(\text{James}, \text{UK})$
$\mathit{takesCourse}(\text{James}, \text{German})$ $\quad$ $\mathit{hasCredits}(\text{German}, 0)$ $\quad$ $\mathit{Country}(\text{Italy})$
$\mathit{Course}(\text{TOC})$ $\quad$ $\mathit{Student}(\text{Luca})$ $\quad$ $\mathit{Country}(\text{UK})$
$\mathit{Course}(\text{ADS})$ $\quad$ $\mathit{Student}(\text{James})$
$\mathit{Course}(\text{German})$

and the $\mathcal{O}_s$ conceptual model from Figure 5.1 into the *DL-Lite*$_R$ ontology

$$\begin{aligned}
\exists \mathit{takesCourse} &\sqsubseteq \mathit{Student} & \exists \mathit{takesCourse}^- &\sqsubseteq \mathit{Course}\\
\exists \mathit{attends} &\sqsubseteq \mathit{Student} & \exists \mathit{attends}^- &\sqsubseteq \mathit{Course}\\
\exists \mathit{comesFrom} &\sqsubseteq \mathit{Student} & \exists \mathit{comesFrom}^- &\sqsubseteq \mathit{Country}\\
\exists \mathit{hasCredits} &\sqsubseteq \mathit{Student} & \exists \mathit{hasCredits}^- &\sqsubseteq \mathit{Integer}\\
\mathit{takesCourse} &\sqsubseteq \mathit{attends} & \mathit{Student} &\sqsubseteq \exists \mathit{attends}\\
\mathit{Student} &\sqsubseteq \exists \mathit{comesFrom}
\end{aligned}$$

in which a role, *hasCredits*, connecting each course to its credit worth is used to capture the attribute. The pair $(\mathcal{O}_s, \mathcal{D}_s)$ constitutes a description logic knowledge base.

Suppose now we want to query $(\mathcal{O}_s, \mathcal{D}_s)$ with φ_s from Example 5.1.2 (i.e., query 5.3). Since the semantics of φ_s over databases is well understood, we might want to reduce OBDAS query evaluation to relational database evaluation by analogy to UCQs. Recall that by Proposition 3.2.5 from Chapter 4 this is possible for UCQs: their (certain) answers over OBDASs can be characterized as their answers over all the databases $\mathcal{D}$ that "comply with" the OBDAS (or knowledge base) $(\mathcal{O}_s, \mathcal{D}_s)$. But this makes no sense in the presence of aggregations. To see why, assume that

$$\mathit{cert}_a(\varphi_s, \mathcal{O}_s, \mathcal{D}_s) := \bigcap \{\mathit{ans}_a(\varphi_s, \mathcal{D}'_s) \mid \mathcal{D}_s \subseteq \mathcal{D}'_s \text{ and } \mathcal{I}(\mathcal{D}'_s) \models \mathcal{O}_s\} \qquad (\dagger)$$

and that we have two databases $\mathcal{D}'_s$ and $\mathcal{D}''_s$ defined by

- $\mathcal{D}'_s := \mathcal{D}_s \cup \{\mathit{comesFrom}(\text{Paolo}, \text{Italy}), \mathit{takesCourse}(\text{Paolo}, \text{TOC})\}$ and
- $\mathcal{D}''_s := \mathcal{D}_s \cup \{\mathit{comesFrom}(\text{Mike}, \text{UK}), \mathit{takesCourse}(\text{Mile}, \text{ADS})\}$.

The databases $\mathcal{D}'_s$ and $\mathcal{D}''_s$ *(i)* contain $\mathcal{D}_s$, *(ii)* satisfy all the domain constraints stated by $\mathcal{O}_s$ and *(iii)* record respectively that a further Italian student, Paolo, attends TOC ($\mathcal{D}'_s$) and a further British student, Mike, attends ADS ($\mathcal{D}''_s$). Thus, in $\mathcal{D}'_s$ two students from Italy and one from the UK are known to attend some course, while in $\mathcal{D}''_s$ the numbers get inverted, and, as a consequence

$$\mathit{ans}_a(\varphi_s, \mathcal{D}'_s) \cap \mathit{ans}_a(\varphi_s, \mathcal{D}''_s) = \{(\text{Italy}, 2), (\text{UK}, 1)\} \cap \{(\text{Italy}, 1), (\text{UK}, 2)\} = \emptyset$$

i.e., the "naive" certain answers of query (5.2) over $(\mathcal{O}_s, \mathcal{D}_s)$ are empty.

In general, for every ATCQ φ and every knowledge base/OBDAS $(\mathcal{O}_s, \mathcal{D}_s)$, applying definition (†) will always yield an empty set of certain answers, since every database compatible with $(\mathcal{O}_s, \mathcal{D}_s)$ may give rise to a *different* group. ♣

To obtain a meaningful notion of certain answers for aggregates over knowledge bases and OBDASs, so that the *same* group arises over each compatible database, we *exploit the notion of core*. The core $\tilde{\varphi}$ of an ATCQ φ is a CQ; therefore, to obtain always the same group(s), it suffices to group over $\tilde{\varphi}$'s certain answers. More precisely, we propose to adopt the following semantics: given an ATCQ φ with aggregation function $\mathbf{agg}$ and a knowledge base/OBDAS $(\mathcal{O}, \mathcal{D})$, we

1. return the certain answers of its core $\tilde{\varphi}$,
2. return the groups these certain answers give rise to and
3. return the value of $\mathbf{agg}$ over each group.

This intuition is captured by the following formal definition:

Definition 5.1.7 (Certain groups and certain answers)**.** Let φ be an ATCQ of grouping variable x and aggregation variable y, and $(\mathcal{O}, \mathcal{D})$ a knowledge base. The *certain group* of tuple c is the bag

$$H_c := \{\!|\sigma(y) \mid c = \sigma(x), \sigma \in \mathit{Sat}^{\mathcal{O}}_{\mathcal{D}}(\tilde{\varphi})|\!\}. \qquad \textbf{(Cgroup)}$$

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

$$\mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D}) := \{(\sigma(x), \mathbf{agg}(H_{\sigma(x)})) \mid \sigma \in \mathit{Sat}^{\mathcal{O}}_{\mathcal{D}}(\tilde{\varphi})\}. \qquad \textbf{(Acert)}$$

Example 5.1.8. Consider again the ATCQ φ_s (query (5.3)). The certain answers semantics for aggregate queries propagates the data in $\mathcal{D}_s$ through the constraints of $\mathcal{O}_s$. In particular, it propagates the tuples in the *takesCourse* relation or table to the *attends* relation. Its core $\tilde{\varphi}_s$ (query (5.4)) gives rise to three satisfying groundings over $(\mathcal{O}_s, \mathcal{D}_s)$, namely: $\sigma := \{x \mapsto \text{Italy}, y \mapsto \text{Luca}, z \mapsto \text{TOC}\}$, $\sigma' := \{x \mapsto \text{Italy}, y \mapsto \text{Luca}, z \mapsto \text{ADS}\}$ and $\sigma'' := \{x \mapsto \text{UK}, y \mapsto \text{James}, z \mapsto \text{ADS}\}$. This gives rise to the certain groups $H_{\text{Italy}} = \{\!|\text{Luca}, \text{Luca}|\!\}$ and $H_{\text{UK}} = \{\!|\text{James}|\!\}$. Since $\mathbf{cntd}$ collapses multiplicities, asking φ_s to $(\mathcal{O}_s, \mathcal{D}_s)$ thus results in

$$\mathit{cert}_a(\varphi_s, \mathcal{O}_s, \mathcal{D}_s) = \{(\text{Italy}, 1), (\text{UK}, 1)\}.$$

Asking instead the $\mathbf{count}$ ATCQ

$$\begin{aligned}\varphi'_s := \{(x, \mathbf{count}(y)) \mid \exists z(&\mathit{Student}(y) \wedge \mathit{attends}(y, z) \wedge \mathit{Course}(z) \wedge \\ &\wedge \mathit{comesFrom}(y, x) \wedge \mathit{Country}(x))\}\end{aligned} \tag{5.5}$$

will yield

$$\mathit{cert}_a(\varphi'_s, \mathcal{O}_s, \mathcal{D}_s) = \{(\text{Italy}, 2), (\text{UK}, 1)\},$$

i.e., Luca is counted twice, because $\mathbf{count}$ is sensitive to multiplicities. In SQL we would have written

```
SELECT cf.CoName,COUNT(cf.SName)
FROM comesFrom cf, attends at
WHERE cf.SName=at.SName)
GROUP BY cf.CoName
```
(5.6)

in place of query (5.2). ♣

Remark 5.1.9. By definition TCQs are special cases of ATCQs. Therefore, for TCQs the notion of aggregate (certain) answers coincides with the notion of (certain) answer: if φ is a TCQ, then, for all ontologies $\mathcal{O}$ and databases $\mathcal{D}$,

$$\mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D}) = \mathit{cert}(\varphi, \mathcal{O}, \mathcal{D}) \quad \text{and} \quad \mathit{ans}_a(\varphi, \mathcal{D}) = \mathit{ans}(\varphi, \mathcal{D})$$

that is, our semantics is a generalization of the standard database and OBDAS semantics to aggregates; moreover,

$$\mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D}) = \bigcap\{\mathit{ans}(\varphi, \mathcal{D}') \mid \mathcal{D} \subseteq \mathcal{D}' \text{ and } \mathcal{I}(\mathcal{D}') \models \mathcal{O}\},$$

as expected.

Note also that aggregate certain answers semantics can also be applied to aggregate conjunctive queries, viz. queries of the form $\{(\bar{x}, \mathbf{agg}(y)) \mid \psi\}$, where ψ is a UCQ, over OBDASs and knowledge bases [CNKT08]. ■

Remark 5.1.10. Considering the core $\tilde{\varphi}$ of an ATCQ φ, has the effect of "closing" the knowledge required for answering φ over a knowledge base $(\mathcal{O}, \mathcal{D})$. Whenever a certain answer for $\tilde{\varphi}$ exists, the variables of $\tilde{\varphi}$ will be instantiated by constants in the active domain $\mathit{adom}(\mathcal{D})$ of $(\mathcal{O}, \mathcal{D})$. This follows from the domain independence property of UCQs (see [AHV95], Chapter 5). In addition, deleting the quantifiers preserves *all* the multiplicities insofar as the same variable(s) can be instantiated possibly many times to the same constant(s) by possibly many pairwise distinct assignments. ■

Finally, we define the associated knowledge base (and database) query answering problem, which, by a slight abuse of notation, we denote also KBQA (resp., QA):

Definition 5.1.11. The *knowledge base query answering* (KBQA) problem (resp. the *database query answering problem* (QA)) for for ATCQs are is the decision problem:

- **Input:** a tuple (c, n), an ATCQ φ and a knowledge base $(\mathcal{O}, \mathcal{D})$ (resp. a database $\mathcal{D}$).
- **Question:** does $(c, n) \in \mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D})$ (resp. $\mathit{ans}_a(\varphi, \mathcal{D})$)?

As before, we are interested in the data complexity of KBQA.

5.2 ATCQ-English

In this section we express ATCQs with the controlled language ATCQ-English, by ascribing *bag types* to controlled language constituents. To stress our use of bag-typed expressions, we make the typing of ATCQ-English constituents explicit. The definition of $\tau(\cdot)$ generalizes the formal semantic analysis of Clifford in [Cli88] and Winter in [MHWB06] and of Karttunen in [Kar77] for English questions and database questions. As was the case for TCQs, we will consider a slightly more expressive counterpart of ATCQs, viz., *graph-shaped aggregate queries*.

Definition 5.2.1 (AGCQs)**.** A *graph-shaped conjunctive aggregate query* is an aggregate query

$$\varphi := \{(x, \mathbf{agg}(y)) \mid \psi\} \tag{AGCQ}$$

of body

$$\psi := \psi_1(x) \wedge r(x, y) \wedge \psi_2(y)$$

where $\varphi_1(x)$ and $\varphi_2(y)$ are GCQs and $r(x, y)$ an atom.

AGCQs contain ATCQs, GCQs and TCQs, but as their bodies are UCQs, the aggregate certain answers (and database) semantics defined in the previous section still applies. We would like our controlled language to express queries like, e.g.,

$$\{\mathbf{max}(n) \mid \exists x(\mathit{Course}(x) \wedge \mathit{hasCredits}(x, n))\} \tag{5.7}$$

with controlled language Wh-question like

Which is the average credit worth of courses? (5.8)

Or queries computing an average

$$\{\mathbf{avg}(n) \mid \exists x(\textit{Course}(x) \wedge \textit{hasCredits}(x, n))\} \tag{5.9}$$

with the question

Which is the average height of courses? (5.10)

in addition to GCQs.

Since Kartunnen in [Kar77], it is customary to associate **HO** meaning representations of type $e{\to}t$ to questions and to assume that they denote the set of their answers. This assumption basically coincides with the database and certain answers semantics of UCQs and their fragments, e.g., GCQs, which are *sets* $\{c_1, \ldots, c_n\}$ of constants. Such was the strategy we followed when defining GCQ-English in Chapter 4.

To generalize this correspondence to AGCQs (and ATCQs) we intend to associate meaning representations of type $e{\to}(\mathbb{Q}{\to}t)$, viz., $e \times \mathbb{Q}{\to}t$, to controlled English questions. This is because the (database or certain answers) semantics of an aggregate query is a set $\{(c_1, n_1), \ldots, (c_k, n_k)\}$ of *ordered pairs* of constants and rational numbers. However, as numbers $n_1, \ldots, n_k$ are the result of applying an aggregation function to a *bag*, a *bag-valued* semantics for English constituents has to be adopted, wherein English words denote bags (the content words) and operations over such bags (the function words).

This makes sense, because a bag-valued semantics can be seen as a generalization of the *set-valued* Boolean semantics of English. Indeed, bag-valued n-ary relations are expressed with typed lambda calculus expressions of type $T_1 \times \cdots \times T_n{\to}\mathbb{N}$, where $\mathbb{N}$ stands (by notation abuse) for the type of non-negative integers, rather than by expressions of $T_1 \times \cdots \times T_n{\to}t$, as would be the case for set-valued n-ary relations. Notice, however, that such bags arise when (and from) answering questions or queries: the data itself does not contain redundancies[3].

Therefore, in order to deal with aggregate questions and queries in controlled English, some criteria must be met:

1. we have to consider a *many-sorted* extension of **HO** where expressions are built using the set of basic types $\{e, t, \mathbb{N}, \mathbb{Q}\}$,
2. such expressions will be interpreted over standard **HO** or **FO** models with possibly *number domains*,
3. we need to understand how (controlled) English can express bags and aggregations, and
4. we need to express *both* aggregate and non-aggregate GCQs.

5.2.1 Expressing Aggregate and Non-Aggregate Queries

Extending HO. Expressions are built using the set $\{e, t, \mathbb{N}, \mathbb{Q}\}$ of basic types. Notice that this implies that logic constants in particular and expressions is general are either polymorphic or overloaded (i.e., interpreted differently according to their typing context).

Conjunction and existential quantification receive now a bag-valued interpretation: the denotation of an expression $\varphi \wedge \psi{:}\mathbb{N}$, w.r.t. a frame $\mathcal{I}$ and an assignment γ will now be

$$(\varphi \wedge \psi)^{\mathcal{I}}_{\gamma} := \varphi^{\mathcal{I}}_{\gamma} \cdot \psi^{\mathcal{I}}_{\gamma},$$

[3] In the relational model (and real-world databases), on which OBDASs are based, database relations contain no repeated tuples.

and of an expression $\exists x^e\varphi$:$\mathbb{N}$,

$$(\exists x^e\varphi)^{\mathcal{I}}_{\gamma} := \sum\{\varphi^{\mathcal{I}}_{\gamma[x:=d]} \mid d \in \mathbb{D}_{\mathcal{I}}\},$$

whereas a $\mathbf{Ho}$ interpretation $\mathcal{I}$ is now said to *satisfy* an expression φ of type $\mathbb{N}$ w.r.t. an assignment γ, in symbols $\mathcal{I}, \gamma \models \varphi$, whenever $\varphi^{\mathcal{I}}_{\gamma} \geq 1$.

Attributes. Aggregation functions are in most cases defined over bags of rational numbers. In ontologies and OBDASs, such numerical values arise from *attributes* (and, by extension, attribute domains), conveyed by special "attribute" roles like *hasHeight* or *hasAge* [CdL$^+$05b]. When combined with a concept instance c, they associate a (possibly unique) rational number n to c. Attributes can be expressed in controlled English with *attribute* constituents like

$$\mathbf{Att} \rightarrow \text{credit worth of} \quad \tau(\mathbf{Att}) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda y^{\mathbb{Q}}.\exists x^e(\mathit{hasCredits}(x,y) \wedge P(x)),$$

that when combined with a bag-typed nominal $\mathbf{N}$ of type $e \rightarrow \mathbb{N}$, give rise to a nominal $\mathbf{N}$ of type $\mathbb{Q} \rightarrow \mathbb{N}$ that denotes a *bag of rational numbers*. Attributes are dealt with also by means of content lexicon entries.

Grouping PPs. Another important feature of ATCQs and AGCQs is grouping, by means of which a quantity obtained from the bag or group of (numerical) attributes of a tuple by applying an aggregation function can be associated to such tuple. In our grammar we have chosen to convey grouping (i.e., `SQL GROUP BY` clauses) by introducing (again in the content lexicon) *grouping prepositional attachments* (grouping $\mathbf{PP}$s) of the form

$$\mathbf{PP} \rightarrow \text{per country} \quad \tau(\mathbf{PP}) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda y^{e}.(P(y) \wedge \mathit{Country}(z) \wedge \mathit{comesFrom}(y,z))$$

which again combine with bag-typed nominals $\mathbf{N}$ to give rise to a bag-typed nominal $\mathbf{N}$.

Notice that they contain a *free* variable z, which will be only abstracted at the root constituent of controlled English questions (and not at any of its dominated constituents), in order to bind together groups and group identifiers (i.e., values of grouping variables).

Aggregate Determiners. Next, we need to define a finite family of distinguished English *aggregate determiners* that express aggregation functions over bags of rational numbers or individuals. Such aggregate determiners, when combined with a bag-typed nominal $\mathbf{N}$, give rise to a meaning representation (and a controlled English constituent) of type $\mathbb{Q}$.

Definition 5.2.2 (Aggregate Determiners)**.** To express SQL aggregation functions we use *aggregate determiners* $\mathbf{Det}$ of global type $(e\rightarrow\mathbb{N})\rightarrow\mathbb{Q}$ or $(\mathbb{Q}\rightarrow\mathbb{N})\rightarrow\mathbb{Q}$:

$$\begin{array}{ll}
\mathbf{Det} \rightarrow \text{the greatest} & \tau(\mathbf{Det}) := \lambda P^{\mathbb{Q}\rightarrow\mathbb{N}}.\mathbf{max}(P) \\
\mathbf{Det} \rightarrow \text{the smallest} & \tau(\mathbf{Det}) := \lambda P^{\mathbb{Q}\rightarrow\mathbb{N}}.\mathbf{min}(P) \\
\mathbf{Det} \rightarrow \text{the total} & \tau(\mathbf{Det}) := \lambda P^{\mathbb{Q}\rightarrow\mathbb{N}}.\mathbf{sum}(P) \\
\mathbf{Det} \rightarrow \text{the number of} & \tau(\mathbf{Det}) := \lambda P^{e\rightarrow\mathbb{N}}.\mathbf{count}(P) \\
\mathbf{Det} \rightarrow \text{the average} & \tau(\mathbf{Det}) := \lambda P^{\mathbb{Q}\rightarrow\mathbb{N}}.\mathbf{avg}(P)
\end{array}$$

Such family can be extended to cover multiplicity insensitive aggregation functions by adding the qualifier "distinct" to the determiner, e.g., "the number of distinct" maps to the expression $\lambda P^{e\rightarrow\mathbb{N}}.\mathbf{cntd}(P)$ of global type $(e\rightarrow\mathbb{N})\rightarrow\mathbb{Q}$.

Non-aggregate GCQs. To express GCQs (and TCQs), or, more, precisely, to capture their set-based (database and certain answers) semantics we rely on the carrier $\xi(\cdot)$ function and on a function $\varsigma(\cdot)$ that "collapses" non-negative integers into Booleans, and turn them into the denotation of the following "overloaded" constant C, by putting, for all interpretations $\mathcal{I}$ and assignments γ,

$$(C(\varphi))^{\mathcal{I}}_{\gamma} := \begin{cases} \xi(\varphi^{\mathcal{I}}_{\gamma}), \text{ if } \varphi{:}e{\to}\mathbb{N}, \\ \varsigma(\varphi^{\mathcal{I}}_{\gamma}), \text{ if } \varphi{:}\mathbb{N}. \end{cases}$$

Such multiplicity-collapsing (and type-casting) operators will be applied when parsing and translating a Y/N- or Wh-question expressing a Boolean or non-Boolean GCQ. It will collapse together repeated occurrences of the same tuple, thus transforming Y/N-question meaning representations of type $\mathbb{N}$ into expressions of type t and Wh-question meaning representations of type $e{\to}\mathbb{N}$ into expressions of type $e{\to}t$.

Bag-Preserving Structural Equivalence. As before, our controlled language will not translate directly into AGCQs and ATCQs, but into (extended) $\mathbf{Ho}$ expressions structurally equivalent to such queries. To this end, we need to extend the notion of structural equivalence introduced in the previous chapter. Two formulas or expressions φ and ψ are said to be *isomorphic* when they are identical up to the renaming of their variables.

Definition 5.2.3 (Structural Equivalence)**.** An (extended $\mathbf{Ho}$) expression $\psi := \lambda x^{e}.\lambda n^{\mathbb{Q}}.n \approx \mathbf{agg}(\lambda y^{e}.\chi(x,y)){:}e{\to}(\mathbb{Q}{\to}t)$ is said to be *structurally equivalent* to an ATCQ or AGCQ $\varphi := \{(x, \mathbf{agg}(y)) \mid \chi'(x,y)\}$, in symbols $\varphi \backsimeq_s \psi$, whenever $\chi'(x,y)$ is isomosphic to $\chi(x,y)$. Similarly, if $\psi := C(\lambda x^{e}.\chi(x))$, $\varphi := \{x \mid \chi'(x)\}$ and $\chi(x)$ is isomosphic to $\chi'(x)$, $\psi \backsimeq_s \varphi$. More in general, for each expression $\varphi := \lambda x^{e}.\chi(x){:}e{\to}T$ and each $\mathbf{Fo}$ formula $\psi := \chi'(x)$, $\varphi \backsimeq_s \psi$ whenever $\chi(x)$ is isomorphic to $\chi'(x)$.

The notion of isomorphism is stronger than the notion of logical equivalence on which the standard previous notion of structural equivalence (i.e., $\equiv_s$) relies. Equivalence, while preserving answers, does not preserve their multiplicities.

Isomorphism does preserve multiplicities, provided that we reason over UCQs as shown by Chaudhuri and Vardi in [CV93], or, more in general, over formulas built using $\exists$, $\wedge$ and $\vee$. It is also a sufficient, but not necessary, condition of logical equivalence. The following proposition follows immediately from this observation.

Proposition 5.2.4. *Let φ and ψ be two $\mathbf{Fo}$ or $\mathbf{Ho}$ formulas built using $\exists$, $\wedge$ and $\vee$. Then $\varphi \backsimeq_s \psi$ implies $\varphi \equiv_s \psi$, but the converse does not hold.*

Later on, in Theorem 5.2.12, we will see that structural equivalence does, indeed, preserve the aggregate database and certain answers semantics of ATCQs and AGCQs.

ATCQ-English. We are now ready to introduce the interrogative controlled language ATCQ-English. As for GCQ-English, we will disregard morphosyntax, since modelling morphosyntactic agreement by means of feature unification is straightforward. We make explicit, instead, co-reference links, co-indexing denoting constituents, in the manner of GCQ-English. Note also that, as discussed above, nominal (i.e., $\mathbf{N}$) constituents, which introduce, together with $\mathbf{VP}$-coordination, recursion in the language, can be modified, in addition to adjectives and relative clauses, by attributes and grouping attachments. See Figure 5.2. In Figure 5.3 the reader will see a function and a content lexicon for the running example.

ATCQ-English does not express ATCQs or AGCQs directly, but relies instead (modulo $\backsimeq_s$) on extended $\mathbf{Ho}$ meaning representations, of which, as we will see later on, ATCQs and AGCQs turn out to be $\mathbf{Fo}$ syntactic sugar.

(Phrase structure rules)	(Semantic actions)
$\mathbf{Q}_{wh} \to \mathbf{Intpro}_i\ \mathbf{N}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := C(\lambda \bar{z}^e.\tau(\mathbf{Intpro}_i)(\tau(\mathbf{N}_i))(\lambda x^e.\tau(\mathbf{S}_{g_i})))$
$\mathbf{Q}_{wh} \to \mathbf{Intpro}_i\ \mathbf{N}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \lambda \bar{z}^e.\tau(\mathbf{Intpro}_i)(\tau(\mathbf{N}_i))(\lambda x^{\mathbb{Q}}.\tau(\mathbf{S}_{g_i}))$
$\mathbf{Q}_{wh} \to \mathbf{Intpro}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := C(\lambda \bar{z}^e.\tau(\mathbf{Intpro}_i)(\lambda x^e.\tau(\mathbf{S}_{g_i}?)))$
$\mathbf{Q}_{wh} \to \mathbf{Intpro}_i\ \mathbf{S}_{g_i}?$	$\tau(\mathbf{Q}_{wh}) := \lambda \bar{z}^e.\tau(\mathbf{Intpro}_i)(\lambda x^{\mathbb{Q}}.\tau(\mathbf{S}_{g_i}?))$
$\mathbf{Q}_{Y/N} \to$ does $\mathbf{NP}_i\ \mathbf{VP}_i?$	$\tau(\mathbf{Q}_{Y/N}) := C(\tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i)))$
$\mathbf{Q}_{Y/N} \to$ is $\mathbf{NP}_i\ \mathbf{VP}_i?$	$\tau(\mathbf{Q}_{Y/N}) := C(\tau(\mathbf{NP}_i)(\tau(\mathbf{VP}_i)))$
$\mathbf{S}_{g_i} \to \mathbf{NP}_{g_i}\ \mathbf{VP}_i$	$\tau(\mathbf{S}_{g_i}) := \tau(\mathbf{NP}_{g_i})(\tau(\mathbf{VP}_i))$
$\mathbf{N}_i \to \mathbf{N}_i\ \mathbf{RelC}_i$	$\tau(\mathbf{N}_i) := \tau(\mathbf{RelC}_i)(\tau(\mathbf{N}_i))$
$\mathbf{N}_i \to \mathbf{N}_i\ \mathbf{PP}$	$\tau(\mathbf{N}_i) := \tau(\mathbf{PP})(\tau(\mathbf{N}_i))$
$\mathbf{N}_i \to \mathbf{Adj}\ \mathbf{N}_i$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Adj})(\tau(\mathbf{N}_i))$
$\mathbf{N}_i \to \mathbf{Att}\ \mathbf{N}_i$	$\tau(\mathbf{N}_i) := \tau(\mathbf{Att})(\tau(\mathbf{N}_i))$
$\mathbf{RelC}_i \to \mathbf{Relp}_i\ \mathbf{S}_{g_i}$	$\tau(\mathbf{RelC}_i) := \tau(\mathbf{Relp}_i)(\lambda x^e.\tau(\mathbf{S}_{g_i}))$
$\mathbf{RelC}_i \to \mathbf{Relp}_i\ \mathbf{S}_{g_i}$	$\tau(\mathbf{RelC}_i) := \tau(\mathbf{Relp}_i)(\lambda x^{\mathbb{Q}}.\tau(\mathbf{S}_{g_i}))$
$\mathbf{VP}_i \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{Adj})$
$\mathbf{VP}_i \to \mathbf{VP}_i\ \mathbf{Crd}\ \mathbf{VP}_i$	$\tau(\mathbf{VP}) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}_i))(\tau(\mathbf{VP}_i))$
$\mathbf{VP}_i \to$ is a $\mathbf{N}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{N}_i)$
$\mathbf{VP}_i \to \mathbf{TV}_{i,i+1}\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{TV}_{i,i+1})(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{VP}_i \to \mathbf{IV}_i$	$\tau(\mathbf{VP}_i) := \tau(\mathbf{IV}_i)$
$\mathbf{VP}_i \to \mathbf{COP}_{i,i+1}\ \mathbf{NP}_{i+1}$	$\tau(\mathbf{TV}_i) := \tau(\mathbf{COP}_{i,i+1})(\tau(\mathbf{NP}_{i+1}))$
$\mathbf{NP}_i \to \mathbf{Det}\ \mathbf{N}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Det})(\tau(\mathbf{N}_i))$
$\mathbf{NP}_i \to \mathbf{Pro}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Pro}_i)$
$\mathbf{NP}_i \to \mathbf{Pn}_i$	$\tau(\mathbf{NP}_i) := \tau(\mathbf{Pn}_i)$
$\mathbf{PP} \to \mathbf{PP}_i\ \mathbf{RelC}_i$	$\tau(\mathbf{PP}) := \tau(\mathbf{PP}_i)(\tau(\mathbf{RelC}_i))$

Figure 5.2: ATCQ-English phrase structure rules. $\mathbf{N}$s are of type $e \to \mathbb{N}$ or $\mathbb{Q} \to \mathbb{N}$. By $\lambda \bar{z}^e$ we denote a (possibly empty) sequence $\lambda x_0^e \cdots \lambda y_n^e$ of abstractions with $\bar{z} \subseteq FV(\lambda x^{\mathbb{Q}}.\tau(\mathbf{S}_{g_i}))$. Polarity, tense, number, gender, etc., features are for the sake of simplicity disregarded.

(Function lexicon)

$\mathbf{Intpro}_i \rightarrow$ which	$\tau(\mathbf{Intpro}_i) := \lambda P^{\mathbb{Q}\rightarrow t}.\lambda x^e.P(x)$
$\mathbf{Intpro}_i \rightarrow$ which	$\tau(\mathbf{Intpro}_i) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda P^{e\rightarrow\mathbb{N}}.\lambda x^e.P(x) \wedge Q(x)$
$\mathbf{Intpro}_i \rightarrow$ who	$\tau(\mathbf{Intpro}_i) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda x^e.P(x)$
$\mathbf{COP}_{i,i+1} \rightarrow$ is	$\tau(\mathbf{COP}_{i,i+1}) := \lambda n^{\mathbb{Q}}.\lambda m^{\mathbb{Q}}.n \approx m$
$\mathbf{Crd} \rightarrow$ and	$\tau(\mathbf{Crd}) := \lambda P^{e\rightarrow\mathbb{N}}\lambda Q^{e\rightarrow\mathbb{N}}.\lambda x^e.(P(x) \wedge Q(x))$
$\mathbf{Relp} \rightarrow$ that	$\tau(\mathbf{Relp}) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda x^e.(P(x) \wedge Q(x))$
$\mathbf{Relp}_i \rightarrow$ who	$\tau(\mathbf{Relp}_i) := \lambda P^{e\rightarrow\mathbb{N}}\lambda x^e.P(x)$
$\mathbf{Det} \rightarrow$ some	$\tau(\mathbf{Det}) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda Q^{e\rightarrow\mathbb{N}}.\exists x^e(P(x) \wedge Q(x))$
$\mathbf{Pro}_i \rightarrow t_i$	$\tau(\mathbf{Pro}_i) := \lambda P^{\mathbb{Q}\rightarrow t}.P(n)$
$\mathbf{Pro}_i \rightarrow$ something	$\tau(\mathbf{Pro}_i) := \lambda P^{e\rightarrow t}.\exists y^e P(y)$
$\mathbf{Pro}_{i-2} \rightarrow$ him	$\tau(\mathbf{Pro}_{i-2}) := \lambda P^{e\rightarrow\mathbb{N}}.P(x)$
$\mathbf{Pro}_{i-1} \rightarrow$ itself	$\tau(\mathbf{Pro}_{i-1}) := \lambda P^{e\rightarrow\mathbb{N}}.P(x)$

(Content lexicon)

$\mathbf{N}_i \rightarrow$ course	$\tau(\mathbf{N}) := \lambda x^e.\mathit{Course}(x)$
$\mathbf{N}_i \rightarrow$ student	$\tau(\mathbf{N}) := \lambda x^e.\mathit{Student}(x)$
$\mathbf{Att} \rightarrow$ credit worth of	$\tau(\mathbf{Att}) := \lambda n^{\mathbb{Q}}.\exists y^e(\mathit{hasCredits}(y,n) \wedge P(y))$
$\mathbf{TV}_{i,i+1} \rightarrow$ study	$\tau(\mathbf{TV}_{i,i+1}) := \lambda\beta^{(e\rightarrow\mathbb{N})\rightarrow\mathbb{N}}.\lambda x^e.\beta(\lambda y^e.\mathit{attends}(x,y))$
$\mathbf{PP} \rightarrow$ per country	$\tau(\mathbf{PP}) := \lambda P^{e\rightarrow\mathbb{N}}.\lambda y^e.(P(y) \wedge \mathit{Country}(z) \wedge \mathit{comesFrom}(y,z))$
$\mathbf{PP}_i \rightarrow$ per country	$\tau(\mathbf{PP}_i) := \lambda Q^{e\rightarrow\mathbb{N}}.\lambda P^{e\rightarrow\mathbb{N}}\lambda y^e.(P(y) \wedge \mathit{Country}(z) \wedge \mathit{comesFrom}(y,z) \wedge Q(z))$
⋮	⋮

Figure 5.3: A sample lexicon for ATCQ-English. We omit aggregate determiners. Note the presence of *grouping* **PP**s and attributes. All words are bag-typed.

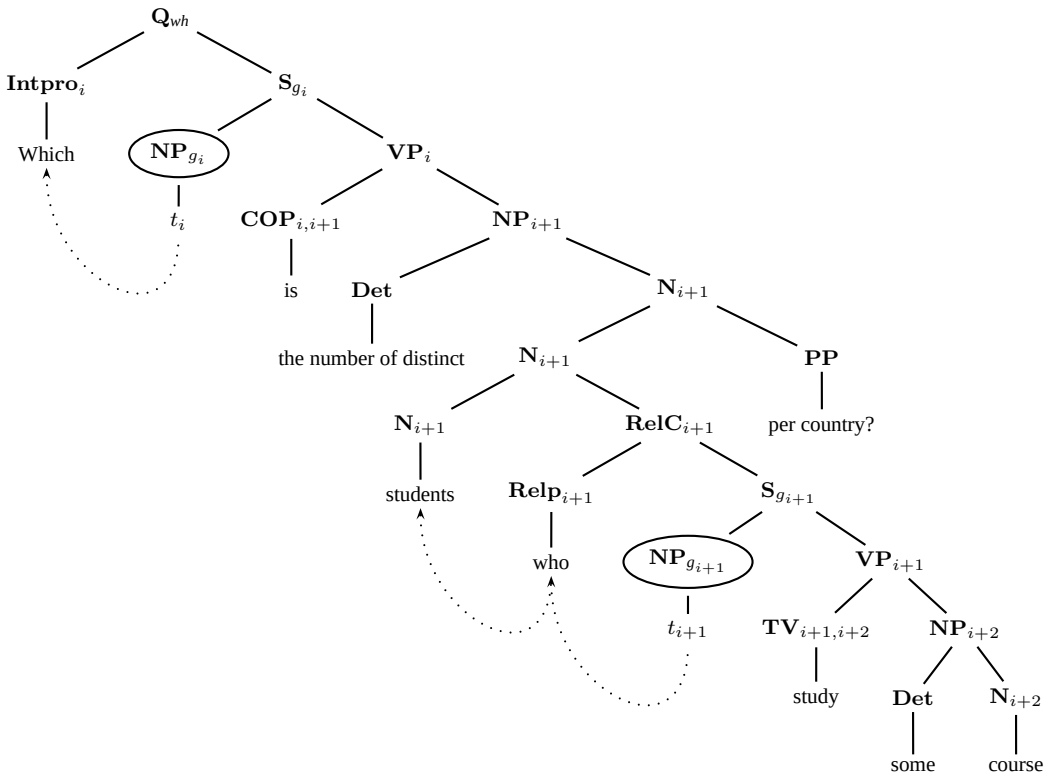


Figure 5.4: Parsing in ATCQ-English.

Theorem 5.2.5. *For every ATCQ-English question Q there exists an AGCQ φ s.t. $\tau(Q) \backsimeq_s \varphi$. Conversely, every AGCQ φ is the image by $\tau(\cdot)$ of some AGCQ-English question Q.*

Proof. ($\Rightarrow$) We need to show that for every Wh-question Q in ATCQ-English there exists an AGCQ φ s.t. $\tau(Q) \backsimeq_s \varphi$. Questions Q are of three kinds, *(i)* aggregate Wh-questions, *(ii)* non-aggregate Wh-questions and *(iii)* (non-aggregate) Y/N-questions. To prove this result, we show something more general, namely that

$$\begin{array}{c}\text{for each } \mathbf{N} \text{ and/or } \mathbf{VP} \text{ constituent of ATCQ-English, there exists}\\ \text{a GCQ } \psi(x) \text{ s.t. } \tau(\mathbf{N}) \backsimeq_s \psi(x) \text{ and/or } \tau(\mathbf{VP}) \backsimeq_s \psi(x)\end{array} \qquad (\dagger)$$

We prove (†) by mutual induction on grammar derivations rooted in $\mathbf{VP}$s and/or $\mathbf{N}$s, taking care that types, polarity and morphosyntactic features, unify. For simplicity, we disregard indexes. It is then easy to see that, for instance, "which is $\mathbf{Det}$ $\mathbf{N}$ per $\mathbf{N}$", where $\mathbf{Det}$ stands for an aggregate determiner, maps to

$$\lambda x^e.\lambda n^{\mathbb{Q}}.n \approx \mathbf{agg}(\lambda y^e.\psi(y) \wedge \psi'(y)){:}e{\rightarrow}(\mathbb{Q}{\rightarrow}t),$$

that "does $\mathbf{NP}$ $\mathbf{VP}$" maps to

$$C(\exists x^e \psi(x)){:}t,$$

or that "which $\mathbf{N}$ $\mathbf{VP}$" maps to

$$C(\lambda x^e.\psi(x)){:}e{\rightarrow}t.$$

($\Leftarrow$) We need to show that for each AGCQ φ there exists a question Q in ATCQ-English s.t. $\tau(Q) \backsimeq_s \varphi$. To prove this, we show, by induction on GCQs $\psi(x)$ rooted in x, that there exists

either a $\mathbf{N}$ or a $\mathbf{VP}$ constituent in ATCQ-English s.t. $\tau(\mathbf{N}) \backsimeq_s \psi(x)$ (resp. $\tau(\mathbf{VP}) \backsimeq_s \psi(x)$):

- **(Basis)** $\psi(x)$ is the image of either "is an A" or "rs himself" or "rs him" or "rs c" or "rs somebody" or "rs somebody who r's him", which are clearly a $\mathbf{N}$ constituent (with possibly $m \leq 0$ subordinated clauses). The other case is analogous.
- **(Inductive step)** If $\psi(x) = \psi'(x) \wedge \psi''(x)$, by IH $\psi'(x)$ is the image of some $\mathbf{N}$ or $\mathbf{VP}$ and similarly for $\psi''(x)$. Hence, $\psi(x)$ has as preimage either "$\mathbf{N\ RelC}$" (where, e.g., $\mathbf{RelC}$ rewrites into the $\mathbf{VP}$ associated to $\psi''(x)$) or "$\mathbf{VP}$ and $\mathbf{VP'}$". The argument is similar for the remaining case.

Clearly then, the AGCQ

$$\{(x, \mathbf{agg}(y)) \mid \psi(x) \wedge r(x, y) \wedge \psi'(y)\},$$

or, more precisely, its $\backsimeq_s$-equivalent $\mathbf{HO}$ meaning representation,

$$\lambda y^e.\lambda n^{\mathbb{Q}}.n \approx \mathbf{agg}(\lambda x^e.\psi(x) \wedge r(x, y) \wedge \psi'(y)){:}e{\to}(\mathbb{Q}{\to}t)$$

will have as preimage in ATCQ-English the question "which is $\mathbf{Det\ N}$ per $\mathbf{N'}$?", where $\mathbf{Det}$ is an aggregate determiner. On the other hand, $\{x \mid \psi(x)\}$ will be the image of "what/who $\mathbf{VP}$?" and $\{\exists x \psi(x)\}$ will be the image of "does anybody $\mathbf{VP}$?" or "is anybody $\mathbf{VP}$?". □

Example 5.2.6. Consider again question Q_s (i.e., question (5.1)). This question belongs to ATCQ-English. The grammar of ATCQ-English gives rise to the parse tree from Figure 5.4. An aggregate determiner is associated to the definite $\mathbf{NP}$ "the number of distinct $\mathbf{N}$" while the *grouping complement* (a $\mathbf{PP}$ attachment) "per $\mathbf{N}$" expresses grouping. We claim that the controlled question (5.1) expresses the ATCQ φ_s from Example 5.1.8 (i.e., query (5.3)).

The value of $\tau(\cdot)$ on the whole question (i.e., the value of $\tau(\cdot)$ on the (root) component $\mathbf{Q}_{wh}$) after λ-application and abstraction and β-normalization is

$$\begin{aligned}\lambda z^e.\lambda m^{\mathbb{Q}}.m \approx \mathbf{cntd}(\lambda x^e.&\mathit{Student}(x) \wedge \exists y^e(\mathit{attends}(x, y)) \wedge \\ &\mathit{Course}(y) \wedge \mathit{comesFrom}(x, z) \wedge \mathit{Country}(z)){:}e{\to}(\mathbb{Q}{\to}t),\end{aligned} \tag{5.11}$$

and that, clearly, $\tau(Q_s) \backsimeq_s \varphi_s$. Notice that in the topmost gapped subordinated sentence, viz., $\mathbf{S}_{g_i}$, two lambda abstractions are performed: *(i)* on the variable $z{:}\ e$, coming all the way down from the grouping $\mathbf{PP}$, and *(ii)* on the variable $k{:}\ \mathbb{Q}$, coming from the trace noun phrase $\mathbf{NP}_{g_i}$. Similarly, question (5.8) expresses ATCQ (5.7). See Figure 5.5. ♣

5.2.2 Adequateness

In this section we show that the structural equivalence $\backsimeq_s$ among ATCQs and AGCQs and ATCQ-English meaning representations does, indeed, preserve both the database and certain answers semantics of aggregations, a condition we term "adequateness". This result strengthens or generalizes Theorem 5.2.5. This is no surprise: $\mathbf{FO}$ is a fragment of $\mathbf{HO}$, so $\mathbf{HO}$ and simply-typed lambda calculus expressions give a more "basic" or fundamental glance on the semantics of queries. Moreover, Libkin et al. have shown in [HLNW99] that query languages with aggregations qre more expressive that $\mathbf{FO}$.

Given a database $\mathcal{D}$ we will denote by $\mathcal{I}'(\mathcal{D}) := (\mathbb{D}_{\mathcal{I}'(\mathcal{D})}, \cdot^{\mathcal{I}'(\mathcal{D})})$ the $\mathbf{HO}$ interpretation induced by $\mathcal{D}$, where:

- $\mathbb{D}_{\mathcal{I}'(\mathcal{D})} \subseteq \mathbf{Dom}$ is a $\mathbf{HO}$ frame of basic domains $\mathit{adom}(\mathcal{D})$ and $\mathbb{Q}$, and

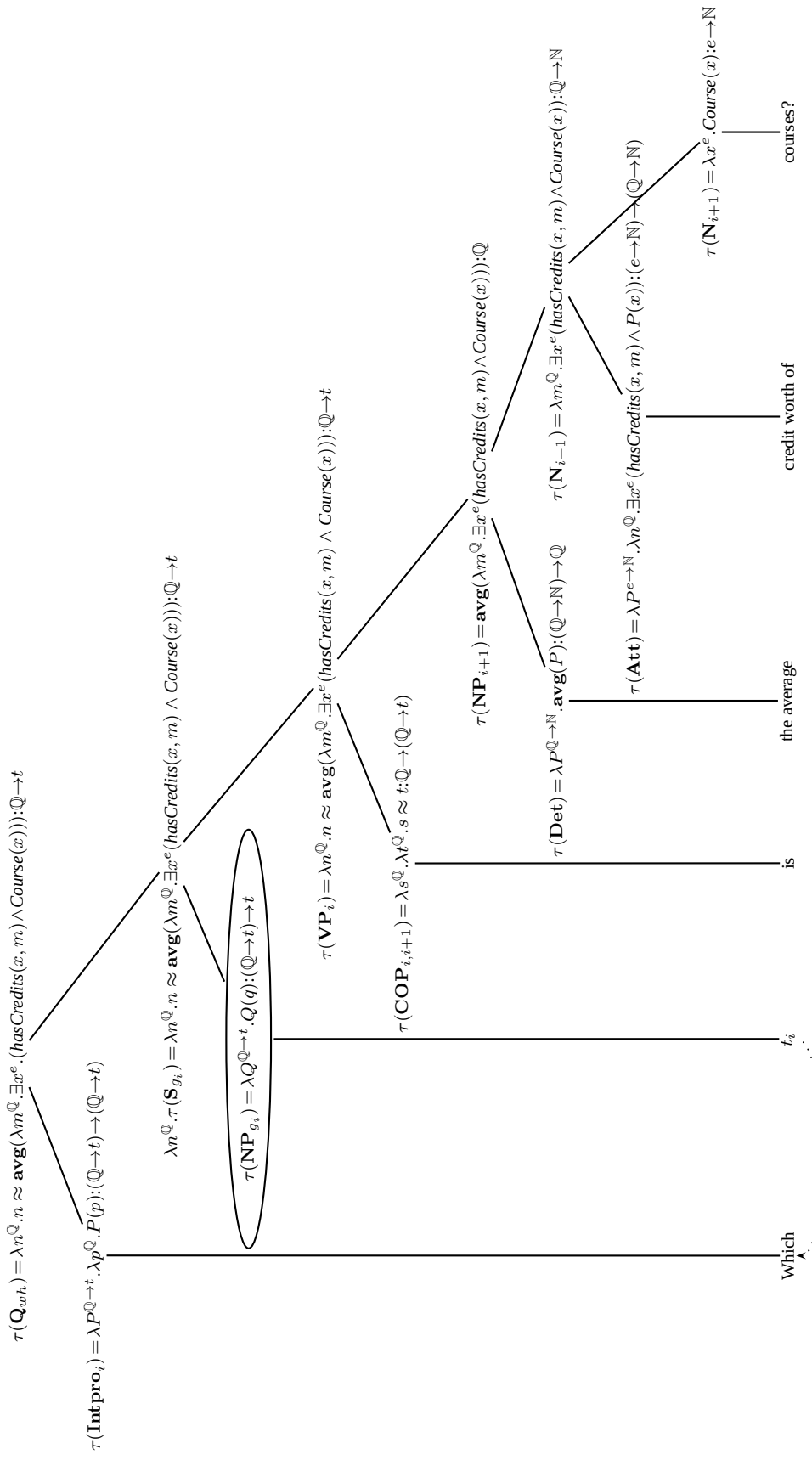


Figure 5.5: Translating "Which is the average credit worth of courses?".

– $\cdot^{\mathcal{I}'(\mathcal{D})}$ maps *(i)* each constant $c \in \mathit{adom}(\mathcal{D})$ to itself, *(ii)* each aggregation function symbol $\mathbf{agg}$ to an aggregation function $\mathbf{agg}^{\mathcal{I}'(\mathcal{D})}$, and *(iii)* each relation symbol S of arity n (hence, in particular, concept names A and role names r) to a characteristic function $S^{\mathcal{I}'(\mathcal{D})}$ where, for each $(c_1, \ldots, c_n) \in \mathit{adom}(\mathcal{D})^n$, $S^{\mathcal{I}'(\mathcal{D})}(c_1, \ldots, c_n) = 1$ iff $S(c_1, \ldots, c_n) \in \mathcal{D}$.

Definition 5.2.7 (Adequateness)**.** We say that an (extended $\mathbf{HO}$) expression $\psi := \lambda x^e.\lambda m^{\mathbb{Q}}.m \approx \mathbf{agg}(\lambda y^e.\chi'(x,y)){:}e{\to}(\mathbb{Q}{\to}t)$ is *adequate* for an ATCQ $\varphi := \{(x, \mathbf{agg}(y)) \mid \chi(x,y)\}$ w.r.t. a database $\mathcal{D}$, whenever, for all tuples (c,n) of constants, $(c,n) \in \mathit{ans}_a(\varphi, \mathcal{D})$ iff for some assignment $\gamma, \mathcal{I}'(\mathcal{D}), \gamma \models m \approx \mathbf{agg}(\lambda y^e.\chi'(x,y)){:}t.$, where $\mathcal{I}'(\mathcal{D})$ is the $\mathbf{HO}$ interpretation induced by database $\mathcal{D}$.

Lemma 5.2.8. *Let φ be an ATCQ, $\mathcal{D}$ a database and ψ an expression such that $\psi \backsimeq_s \varphi$. Then ψ is adequate for ψ w.r.t. $\mathcal{D}$.*

Proof. Let φ, ψ and $\mathcal{D}$ be as in the statement of the Lemma. Let (c,n) be a tuple of individual and numerical constants. We need to prove that

$$(c,n) \in \mathit{ans}_a(\varphi, \mathcal{D}) \quad \text{iff} \quad \text{for some } \gamma, \mathcal{I}'(\mathcal{D}), \gamma \models m \approx \mathbf{agg}(\lambda y^e.\chi'(x,y)){:}t. \qquad (\dagger)$$

($\Rightarrow$) If $(c,n) \in \mathit{ans}_a(\varphi, \mathcal{D})$, then, there exists γ s.t. $\gamma(x) = c$, $n = \mathbf{agg}(G_{\gamma(x)})$, and $\mathcal{I}(\mathcal{D}), \gamma'' \models \chi(x,y)$, for all γ'' s.t. $\gamma''(x) = \gamma(x)$. Let $\mathcal{I}'(\mathcal{D})$ be the $\mathbf{HO}$ counterpart of $\mathcal{I}(\mathcal{D})$. Define γ' over $\mathcal{I}'(\mathcal{D})$ from γ by putting $\gamma'(x) := \gamma(x)$ and $\gamma'(m) := n$. Now, recall that $\varphi \backsimeq_s \psi$. Multiplicities are preserved modulo this condition by γ' and $\mathcal{I}'(\mathcal{D})$, since, indeed

$$\begin{aligned}
\gamma'(m) &= \mathbf{agg}(G_{\gamma(x)}) \\
&= \mathbf{agg}(\{\!|\gamma''(y) \mid \gamma''(x) = \gamma(x), \mathcal{I}(\mathcal{D}), \gamma'' \models \chi(x,y)|\!\}) \\
&= \mathbf{agg}^{\mathcal{I}'(\mathcal{D})}(\{\!|\gamma'''(y) \mid \gamma'''(x) = \gamma'(x), \mathcal{I}'(\mathcal{D}), \gamma''' \models \chi'(x,y){:}t|\!\}) \\
&= \mathbf{agg}^{\mathcal{I}'(\mathcal{D})}((\lambda y^e.\chi(x,y){:}e{\to}\mathbb{N})^{\mathcal{I}'(\mathcal{D})}_{\gamma'}),
\end{aligned}$$

and, as a result, $\mathcal{I}'(\mathcal{D}), \gamma' \models m \approx \mathbf{agg}(\lambda y^e.\chi(x,y)){:}t$ as desired.

($\Leftarrow$) Let $\mathcal{I}'(\mathcal{D}), \gamma' \models m \approx \mathbf{agg}(\lambda y^e.\chi(x,y))$ be as in the statement of the claim. Let $\mathcal{I}(\mathcal{D})$ be the $\mathbf{FO}$ counterpart of $\mathcal{I}'(\mathcal{D})$. Define an assignment $\gamma \in \mathit{Sat}_{\mathcal{D}}(\tilde{\varphi})$ by exploiting γ', viz., by putting $\gamma(x) := \gamma'(x)$. Such assignment preserves multiplicities and groups, since, modulo $\varphi \backsimeq_s \psi$,

$$\begin{aligned}
n &= \mathbf{agg}^{\mathcal{I}'(\mathcal{D})}((\lambda y^e.\chi(x,y){:}e{\to}\mathbb{N})^{\mathcal{I}'(\mathcal{D})}_{\gamma'}) \\
&= \mathbf{agg}^{\mathcal{I}'(\mathcal{D})}(\{\!|\gamma'''(y) \mid \gamma'''(x) = \gamma'(x), \mathcal{I}'(\mathcal{D}), \gamma''' \models \chi'(x,y){:}t|\!\}) \\
&= \mathbf{agg}(\{\!|\gamma''(y) \mid \gamma''(x) = \gamma(x), \mathcal{I}(\mathcal{D}), \gamma'' \models \chi(x,y)|\!\}) \\
&= \mathbf{agg}(G_{\gamma(x)})
\end{aligned}$$

and $(c,n) \in \mathit{ans}_a(\varphi, \mathcal{D})$ as desired. □

As we observed in Remark 5.1.10, the domain-independence of UCQs implies that the variables of ATCQs and AGCQs will be instantiated to points in the active domain of knowledge bases. Clearly, we must be sure that the structural equivalence $\backsimeq_s$ relation preserves domain-independence, in addition to multiplicities.

Domain-independence can be formally defined through the notion of relativization. Given a set $D \subseteq \mathbf{Dom}$ we define the *relativization* to D of an expression u by structural recursion on lambda-expressions as follows

$$\begin{aligned}
\mathit{rel}(c, D) &:= c \\
\mathit{rel}(x, D) &:= \top_D(x) \\
\mathit{rel}(u(u'), D) &:= \mathit{rel}(u, D)(\mathit{rel}(u', D)) \\
\mathit{rel}(\lambda x^T.u, D) &:= \lambda x^T.\mathit{rel}(u, D)
\end{aligned}$$

where $\top_D(\cdot)$ is the function or constant s.t., for all $d \in \mathbf{Dom}$ and all interpretations $\mathcal{I}$,

$$\top_D^{\mathcal{I}}(d) := \begin{cases} d, \text{ if } d \in D, \\ 0, \text{ otherwise.} \end{cases}$$

We say that an expression u is *domain-independent*, iff, for all interpretations $\mathcal{I} = (\mathbb{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$, all sets $\mathbb{D}_{\mathcal{I}} \subseteq D \subseteq \mathbf{Dom}$ and all substitutions $\sigma\colon FV(u) \to \mathbf{Dom}$, $u\sigma^{\mathcal{I}} = \mathit{rel}(u, D)^{\mathcal{I}}\sigma$. Clearly, structural equivalece preserves this property (and hence applies to ATCQ-English semantic representations).

Proposition 5.2.9. *If ψ is an extended $\mathbf{HO}$ expression and φ an ATCQ or an AGCQ s.t. $\varphi \backsimeq_s \psi$, then ψ is domain-independent.*

Modulo domain-independence, we can now show that structural equivalence preserves not only aggregate database answers, but also aggregate certain answers.

Definition 5.2.10 (Adequateness)**.** We say that an (extended $\mathbf{HO}$) expression $\psi := \lambda x^e.\lambda m^{\mathbb{Q}}.m \approx \mathbf{agg}(\lambda y^e.\chi'(x,y))\!:\!e{\to}(\mathbb{Q}{\to}t)$ is *adequate* for an ATCQ $\varphi := \{(x, \mathbf{agg}(y)) \mid \chi(x,y)\}$ w.r.t. a knowledge base $(\mathcal{O}, \mathcal{D})$, whenever, for all tuples (c,n) of constants, $(c,n) \in \mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D})$ iff $\mathcal{O} \cup \mathcal{D} \models \psi(c)(n)\!:\!t$.

Lemma 5.2.11. *Let φ be an ATCQ, $(\mathcal{O}, \mathcal{D})$ a knowledge base and ψ an expression such that $\psi \backsimeq_s \varphi$. Then ψ is adequate for φ w.r.t. $(\mathcal{O}, \mathcal{D})$.*

Proof. Let φ, ψ and $(\mathcal{O}, \mathcal{D})$ be as in the statement of the Lemma. Let (c,n) be a tuple of individual and numerical constants. We need to prove that

$$(c,n) \in \mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D}) \quad \text{iff} \quad \mathcal{O} \cup \mathcal{D} \models \psi(c)(n)\!:\!t. \tag{$\dagger$}$$

($\Rightarrow$) Given an arbitrary $\mathcal{D} \subseteq \mathcal{D}'$, let $\mathcal{I}'(\mathcal{D}')$ be an arbitrary $\mathbf{HO}$ model of $(\mathcal{O}, \mathcal{D})$. Given an arbitrary satisfying substitution σ, it is easy to see using an argument analogous to those employed for Lemma 5.2.8, that, modulo $\varphi \backsimeq_s \psi$, for every arbitrary assignment γ' over $\mathcal{I}'(\mathcal{D}')$ such that $\gamma'(x) := \sigma(x)$, $\gamma'(m) = \mathbf{agg}(H_{\sigma(x)})$, i.e., multiplicities are preserved. Hence, since γ' was arbitrary, $\mathcal{I}'(\mathcal{D}') \models \psi(c)(n)\!:\!t$ and a fortiori $\mathcal{O} \cup \mathcal{D} \models \psi(c)(n)\!:\!t$.

($\Leftarrow$) We know that $(\mathcal{O}, \mathcal{D}) \models \psi(c)(n)\!:\!t$. This means that for all $\mathcal{D} \subseteq \mathcal{D}'$ s.t. $\mathcal{I}'(\mathcal{D}') \models \mathcal{O}$ and $\mathcal{I}'(\mathcal{D}') \models \psi(c)(n)$,

$$n^{\mathcal{I}'(\mathcal{D}')} = \mathbf{agg}^{\mathcal{I}'(\mathcal{D}')}((\lambda y^e.\chi'(c^{\mathcal{I}'(\mathcal{D}')}, y)\!:\!e \to \mathbb{N})^{\mathcal{I}'(\mathcal{D}')}), \tag{$\ddagger$}$$

where $c^{\mathcal{I}'(\mathcal{D}')} \in \mathit{adom}(\mathcal{D})$. Let $\mathcal{I}(\mathcal{D}')$ be an arbitrary $\mathbf{FO}$ model of $(\mathcal{O}, \mathcal{D})$ derived from $\mathcal{I}'(\mathcal{D}')$. Let σ be the substitution such that $\sigma(x) := c^{\mathcal{I}'(\mathcal{D}')}$. Since $\chi(x,y) \backsimeq_s \chi'(x,y)$ by hypothesis, this together with ($\ddagger$) implies that *(i)* $\sigma \in \mathit{Sat}_{\mathcal{D}}^{\mathcal{O}}(\tilde{\varphi})$, *(ii)* $H_{\sigma(x)} = (\lambda y^e.\chi'(\sigma(x), y)\!:\!e{\to}\mathbb{N})$ and, ultimately, *(iii)* $(c,n) \in \mathit{cert}_a(\varphi, \mathcal{O}, \mathcal{D})$. □

From Lemmas 5.2.8 and 5.2.11 we immediately derive the "adequateness" theorem.

Theorem 5.2.12. *Let φ be an ATCQ, $\mathcal{O}$ an ontology, $\mathcal{D}$ a database and ψ an (extended $\mathbf{HO}$) expression such that $\psi \backsimeq_s \varphi$. Then ψ is adequate for φ w.r.t. both $\mathcal{D}$ and $(\mathcal{O}, \mathcal{D})$.*

Interestingly, this theorem substantiates the strategy followed in this chapter and, it can be claimed, in [CNKT08], when we defined the notion of certain answers, viz., to aggregate on top of the certain answers of a core. Furthermore, it generalizes to AGCQs and aggregate UCQs. By considering AGCQs, it immediately follows that ACTQ-English does indeed capture exactly AGCQs.

First Reading	Second Reading
$\{\mathbf{count}(x) \mid \exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(y,z))\}$	$\{\mathbf{count}(x) \mid \exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(x,z))\}$

`SELECT COUNT(lo.MName)`
`FROM loves lo,hasChild hc`
`WHERE lo.WName=hc.WName`
	`SELECT COUNT(lo.MName)`
	`FROM loves lo`
	`WHERE EXISTS (SELECT *`
	`  FROM hasChild hc`
	`  WHERE lo.WName=hc.WName)`

$\exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(y,z))$	$\exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(y,z))$

`SELECT lo.MName`
`FROM loves lo,hasChild hc`
`WHERE lo.WName=hc.WName`
	`SELECT lo.MName`
	`FROM loves lo`
	`WHERE EXISTS (SELECT *`
	`  FROM hasChild hc`
	`  WHERE lo.WName=hc.WName)`

Table 5.2: The ambiguity of query (5.12) affects groups, but is harmless for query (5.13).

5.3 Bags, Nested Queries and Ambiguity

Although ATCQ-English is a controlled language where each question is compositionally mapped by $\tau(\cdot)$ to a *unique* (up to structural equivalence) ATCQ or AGCQ, some residual ambiguity might still arise from the formal queries themselves. ATCQs are an abstract, logic-based notation for SQL queries where *(i)* existential quantifiers and *(ii)* conjunctions in positive conjunctive bodies model each *two* different SQL constructs (see [EN04], Chapter 8):

- existential quantifiers stand both for SQL *nested existential conditions* and table projection (the π operator of relational algebra);
- conjunction stands both for boolean conjunction and table natural join (the $\bowtie$ operator of relational algebra).

These different interpretations affect multiplicities in groups: SQL *nested* existential conditions or queries evaluate to true as soon as a tuple verifies them, disregarding its multiplicity. As such, they are often used instead of the `DISTINCT` operator to collapse multiplicities in aggregate queries. But **FO**-based notations cannot capture this subtlety.

Example 5.3.1. Consider Table 5.2. The **count** ATCQ

$$\{\mathbf{count}(x) \mid \exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(y,z))\} \tag{5.12}$$

that asks for the number of men who love women with daughters. It gives to *two* SQL queries, one of which (the second reading) contains a nested condition or query. The same holds for the non-aggregate core TCQ

$$\exists y(\mathit{loves}(x,y) \wedge \exists z\, \mathit{hasChild}(y,z)) \tag{5.13}$$

that asks for the men who love women with daughters and that gives also rise to two SQL queries. Consider now the following family database $\mathcal{D}_f$, where f stands for "family",

loves

MName	WName
John	Laura

hasChild

WName	CName
Laura	Sara
Laura	Sandra

The nested SQL query counts joins *only once*, whereas the non-nested SQL counts *all* joins, thus affecting the multiplicity of John. Under the first reading, we get $G_{()} = \{\!|\text{John}, \text{John}|\!\}$, and hence as answer $\{2\}$, whereas under the second reading, we get instead $G_{()} = \{\!|\text{John}|\!\}$ and hence as answer $\{1\}$.

This problem does not arise with the readings of the TCQ, because the semantics of TCQs (and, more in general, of UCQs) is *set based*. Under both readings we get as answer the singleton $\{\text{John}\}$, i.e., they are *equivalent*. ♣

Notice that this ambiguity is directly linked to the bag semantics of aggregate queries. It can be prevented by ruling out the nested readings of ATCQs and AGCQs. With this proviso, ATCQ-English questions can be mapped not only to a unique ATCQ or AGCQ, but also to a *unique* SQL query.

5.4 Comparisons, $\vee$, $\neg$ and $\forall$

In some question corpora[4], questions with *comparisons*, *negation* and *universal quantification*, also occur rather frequently. For instance, in a corpus of questions issued to the U.S. National Library of Medicine web interface[5] one can observe that more than 40% contain negation(s) and/or disjunction(s) and that more than 15% contain universal quantification(s).

In natural language, comparisons are expressed by comparatives (both majorative and diminutive) and equative adjectives, to be interpreted over some totally ordered domain. Disjunctions are expressed by the coordinating particle "or". Negation is expressed by the negative particle "not" (alone or in combination with an auxiliar verb such as "does"). Universal quantification is expressed by many different $\mathbf{Det}$s such as "every", "all", "no" and "only". Examples of English questions consistent with the student domain of our running example which contain such lexical entries are:

Which course is harder than (strictly harder than, as hard as) ADS? (5.14)

Which student does not attend ADS? (5.15)

Which student attends only courses harder than ADS? (5.16)

Where by "harder than" we may mean (to follow, again, the running example) that a course has more credits (credits provide a measure of a course's relevance and difficulty). Cleary, since real users might use such questions, it makes sense to ask whether adding comparisons, negation and universal quantification give rise to tractable or to intractable data complexity.

The strategy we propose is extending the syntax and semantics of ATCQs and studying the computational properties of the resoluting knowledge base query answering problem (i.e., of KBQA).

In SQL, universal quantifiers, comparisons and negation typically occur in the `WHERE` and `HAVING` clauses, viz., as (complex) Boolean selection conditions which are used as "filters" over joins of relations, that is, as conditions that filter away from such joins those tuples that do not satisfy the Boolean selection condition. As such, they are essentially set-valued and thus cannot be per se subject to the bag typing and the bag semantics outlined for ATCQs and AGCQs. To make explicit this implicit typing, we enrich on the one hand the syntax of queries and modify on the other hand the notion of core, so that these new constructs do not give rise to multiplicities.

Definition 5.4.1 (Extended aggregate tree shaped query)**.** An *extended* ATCQ is as a query of the form

$$\varphi := \{(x, \mathbf{agg}(y)) \mid \psi_1(x) \wedge r(x, y) \wedge \psi_2(y)\}$$

where the tree-shaped query $\psi_1(x)$ (resp. $\psi_2(y)$) is a conjunction

$$\psi_1(x) := \psi_{1,b}(x) \wedge \psi_{1,s}(x)$$

of a *bag condition* $\psi_{1,b}(x)$ and a (set-valued) *selection condition* $\psi_{1,s}(x)$. The core $\tilde{\varphi}$ of such extended ATCQs is now defined as the underlying TCQ obtained by deleting all the quantifiers that *do not occur* within its selection conditions $\psi_{1,s}(x)$ and $\psi_{2,s}(y)$.

The notions of aggregate database answers and aggregate certain answers are left unchanged: by leaving the quantifiers in the selection conditions of cores, multiple instantiations of variables under different $\mathbf{Fo}$ assignments are disregarded.

[4] This observations stem from joint work with E. Bonin, D. Carbotta, R. Bernardi and D. Calvanese on answering natural language questions over the QUONTO OBDAS using wide coverage combinatorial grammar-based statistical parsersing in [BBC+07].

[5] `http://gateway.nlm.nih.gov`

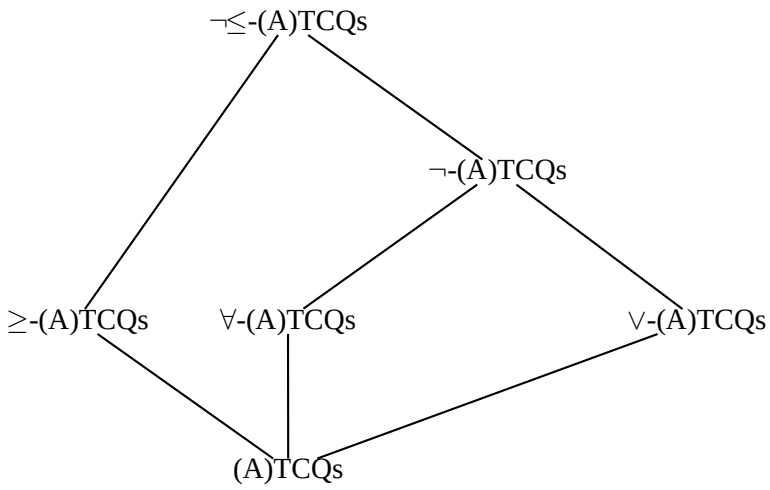


Figure 5.6: Expressive power of the extensions of (A)TCQs.

By default, set conditions are TCQs, and hence built using existential quantification and conjunction over unary and binary atoms, in which case no real expressivity is added to (A)TCQs. By contrast, expressiveness increases if we *close* TCQs under negation, disjunctions or universal restrictions, or if we add comparisons, by stipulating that

- if $\psi(x)$ is a selection condition, so is $\neg\psi(x)$ ($\neg$-(A)TCQs) ,
- if $\psi(x)$ and $\psi(x)'$ are selection conditions, so is $\psi(x) \vee \psi'(x)$ ($\vee$-(A)TCQs),
- if $\psi(x)$ is a selection condition, $\forall y(R(x,y) \Rightarrow \psi(y))$ ($\forall$-(A)TCQs), and
- selection conditions can contain atoms $x\,\theta\,y$, for $\theta \in \{\leq, \geq, <, >, \approx\}$ ($\geq$-(A)TCQs).

Figure 5.6 shows the resulting lattice. Notice that universal restriction can be defined in terms of negation and existential quantification and that $\forall$-(A)TCQs are contained in $\neg$-(A)TCQs.

Extending the controlled interrogative language is a much harder issue, though. In languages such as English it is unclear under which conditions the function words “or”, “only” and “not” receive a bag-valued as opposed to a set-valued interpretation, or under which set-valued constituents combine with bag-valued constituents. We will therefore not deal with this issue in this thesis, but will proceed under the assumption that expressing extended ATCQs in (controlled) English is possible. This assumption is reasonable because, in the context of formal query languages and computational semantics, set semantics is a special case of bag semantics.

Notice that since $\vee$-TCQs, $\leq$-TCQs, $\neg$-TCQs, $\forall$-TCQs and $\neg\leq$-TCQs are restricted kinds of first order queries, we will repeatedly invoke *(i)* the definition of certain answers for UCQs and first order queries and *(ii)* Proposition 3.2.5 when deriving data complexity lower and upper bounds. This is due to the fact that certain answers are based on $\mathbf{FO}$ entailment and can be immediately generalized (or extended) to arbitrary $\mathbf{FO}$ formulas (see [AHV95], Chapter 19).

5.5 Data Complexity

In this section we show that adding $\forall$, $\leq$ and $\neg$ to conditions make query answering hard. The $\vee$ operator alone, however, does not [CdL$^+$06]. We give complexity upper bounds for the *most expressive* languages and lower bounds for the *least expressive* languages. We consider as “ontology languages” the following controlled languages seen thus far: Lite-English, COP and COP+TV. Note that “only” and “or” in questions (and queries) are expressible as soon as we add “not” to ATCQ-English (and hence close ATCQ bodies under negations). Table 5.4 summarizes the main

Algorithm 1 Deciding the aggregate certain answers of $\mathbf{sum}$-TCQs

```
1: procedure CERT_sum(φ, O, D, c, n)
2:     φ̃ ← CORE(φ);
3:     for σ ∈ Sat_O^D(φ̃) do
4:         n_σ(x) ← 0;
5:         for σ′ ∈ Sat_O^D(φ̃) do
6:             if σ′(y) = σ(y) then
7:                 n_σ(x) ← n_σ(x) + σ(y);
8:                 if (c, n) = (σ(x), n_σ(x)) then
9:                     return true;
10:                end if
11:            end if
12:        end for
13:    end for
14:    return false;
15: end procedure
```

data complexity results shown in this section.

In what follows, we will derive complexity lower bounds by defining reductions from the **NPTime**-complete satisfiability problem for *2+2 formulas* studied by Scharf in [Sch93] (whose proof we repeatedly adapt).

Definition 5.5.1. The satisfiability problem for *propositional 2+2 formulas* (2+2-SAT) is the decision problem defined by

- **Input:** a propositional formula in conjunctive normal form $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ where each conjunct $\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$ is a disjunction of two positive (non-negated) and two negative (negated) propositional atoms.
- **Question:** does there exist a truth assignment $\delta(\cdot)$ s.t. $\delta(\psi) = 1$?

Interestingly, answering aggregate queries can be reduced to answering their cores. To check whether a tuple (c, n) is a certain answer to a (possibly extended) ATCQ $\varphi := \{(x, \mathbf{agg}(y)) \mid \psi\}$ over a knowledge base $(\mathcal{O}, \mathcal{D})$, in general, we *(i)* check whether x is instantiated to c by a certain grounding $\sigma \in \mathit{Sat}_{\mathcal{D}}^{\mathcal{O}}(\tilde{\varphi})$ and then *(ii)* loop over the (finitely many) certain groundings $\sigma' \in \mathit{Sat}_{\mathcal{D}}^{\mathcal{O}}(\tilde{\varphi})$ for y, updating at each step the value of $\mathbf{agg}$ on the group H_c, until $\mathbf{agg}$ returns n. Otherwise, our procedure will return a negative answer. The data complexity of answering φ thus depends on computing $\mathit{Sat}_{\mathcal{D}}^{\mathcal{O}}(\tilde{\varphi})$ and is bounded above by the data complexity of answering its core $\tilde{\varphi}$. Updating the value of $\mathbf{agg}$ adds a $\mathbf{O}(\mathit{adom}(\mathcal{D}))$ space overhead on top of the certain assignments $\mathit{Sat}_{\mathcal{D}}^{\mathcal{O}}(\tilde{\varphi})$ computation.

This procedure is spelled out for $\mathbf{sum}$-queries by Algorithm 1, where the imbricated for-loops compute, essentially, the certain answers of the core, viz., in the outer loop, the answers that contribute to comouting group identifiers and, in the inner loops, the answers that give rise to groups. Notice that when computing a sum, we add rational numbers. The resulting quantity can therefore be any positive or negative fraction. By making some minor modifications, the same algorithm can be used for $\mathbf{max}$, $\mathbf{min}$, $\mathbf{count}$ and $\mathbf{avg}$-queries. Notice also that storing the current value $n_{\gamma(x)}$ of the sum requires at most $\mathbf{O}(\mathit{adom}(\mathcal{D}))$ space.

5.5.1 $\vee$-(A)TCQs

Theorem 5.5.2. *Answering $\vee$-ATCQs over Lite-English and COP knowledge bases is in* **LSpace** *in data complexity.*

Proof. Lite-English expresses the *DL-Lite* ontology languages (Theorems 4.1.4–4.1.7). Moreover, it contains COP (Theorem 4.1.15). On the other hand, we can reduce answering an $\vee$-ATCQ φ to answering its core $\tilde{\varphi}$. The core $\tilde{\varphi}$ is a $\vee$-TCQ, hence a UCQ. The result then follows from the data complexity of answering UCQs over *DL-Lite* knowledge bases (Theorem 3.3.5). □

Theorem 5.5.3. *Answering $\vee$-ATCQs over COP+TV knowledge bases is in* **PTime** *in data complexity.*

Proof. We reason as before, noting that later (see Chapter 7) in this thesis we show (Theorem 7.5.4) that answering UCQs over COP+TV knowledge bases is in **PTime** in data complexity. □

5.5.2 $\neg$- and $\forall$-(A)TCQs

To obtain a *data complexity* reduction from 2+2-SAT, we encode 2+2 formulas ψ into *databases* $\mathcal{D}_\psi$. Thereafter, by considering a *fixed* ontology $\mathcal{O}$ and a *fixed* TCQ φ, we show that the computation of the certain answers can be used as an algorithm that checks for ψ's satisfiability.

Theorem 5.5.4. *Answering $\forall$-TCQs over knowledge bases $(\mathcal{O}, \mathcal{D})$ where $\mathcal{O}$ contains disjointness assertions is* **coNPTime***-hard in data complexity. It is in* **coNPTime** *for COP+TV, Lite-English, COP and $\forall$-ATCQs.*

Proof. **(Hardness)** By reduction from 2+2-SAT. A *disjointness* assertion is a description logic assertion of the form $C \sqsubseteq \neg C'$ (that states that concepts C and C' have a disjoint denotation). Let $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ be a 2+2 formula over the propositional atoms $At(\psi) := \{l_1, \ldots, l_m\}$ with, for $i \in [1, k]$,

$$\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}.$$

Consider the role names pos_1, pos_2, neg_1 and neg_2, the concept names A_f and A_t, and the *attribute role hasValue*. The reduction proceeds as follows:

– Map ψ to the database $\mathcal{D}_\psi$:

$$\{pos_1(c_1, p_{11}), pos_2(c_1, p_{12}), neg_1(c_1, n_{11}), neg_2(c_1, n_{12}),$$
$$\vdots$$
$$pos_1(c_k, p_{k1}), pos_2(c_k, p_{k2}), neg_1(c_k, n_{k1}), neg_2(c_k, n_{k2}), A_t(true)\}.$$

– Consider the ontology

$$\mathcal{O} := \{A_f \sqsubseteq \neg A_t\}.$$

– Consider the (boolean) $\forall$-TCQ query

$$\begin{aligned}\varphi \;:=\; & \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (pos_1(x, y_1) \wedge \forall z_1 (hasValue(y_1, z_1) \Rightarrow A_f(z_1)) \wedge pos_2(x, y_2) \wedge \\ & \forall z_2 (hasValue(y_2, z_2) \Rightarrow A_f(z_2)) \wedge neg_1(x, y_3) \wedge \exists z_3 (hasValue(y_3, z_3) \wedge A_t(z_3)) \wedge \\ & neg_2(x, y_4) \wedge \exists z_4 (hasValue(y_4, z_4) \wedge A_t(z_4))).\end{aligned}$$

We claim that

$$() \notin \mathit{cert}(\varphi, \mathcal{O}, \mathcal{D}_\psi) \quad \text{iff} \quad \psi \text{ is satisfiable.} \tag{†}$$

($\Leftarrow$) If ψ is satisfiable, then there exists a truth assignment $\delta\colon \mathit{At}(\psi) \to \{0,1\}$ s.t. $\delta(\psi) = 1$. Construct an interpretation of $\mathcal{D}_\psi$ that is a model of $\mathcal{O}$ as follows. Consider a database $\mathcal{D} \supseteq \mathcal{D}_\psi$. Clearly, $\mathcal{I}(\mathcal{D}) \models \mathcal{D}_\psi$. Next, pick a $v \in \mathit{adom}(\mathcal{D}_\psi)$ and put, for $l \in \mathit{At}(\psi)$,

$$(l, v) \in \mathit{hasValue}^{\mathcal{I}(\mathcal{D})} \text{ and } v \in A_t^{\mathcal{I}(\mathcal{D})} \quad \text{iff} \quad \delta(l) = 1.$$

One can see that $\mathcal{I}(\mathcal{D})$ is as desired and that, for all variable assignments γ, $\mathcal{I}(\mathcal{D}), \gamma \not\models \varphi$, i.e., that $() \notin \mathit{cert}(\varphi, \mathcal{O}, \mathcal{D}_\psi)$.

($\Rightarrow$) If the certain answers are empty, then there exists a interpretation $\mathcal{I} \models (\mathcal{O}, \mathcal{D}_\psi)$, s.t., $\mathcal{I} \not\models \varphi$. Define now a truth assignment $\delta\colon \mathit{At}(\psi) \to \{0,1\}$ by putting, for all $l \in \mathit{At}(\psi)$,

$$\delta(l) = 1 \quad \text{iff} \quad \text{for some } v,\ (p, v) \in \mathit{hasValue}^{\mathcal{I}} \text{ and } v \in A_t^{\mathcal{I}}.$$

Clearly, $\delta(\psi) = 1$, i.e., ψ is satisfiable. Claim (†) follows.

Intuitively, the soundness and the completeness of the reduction come from the fact that we can "simulate" boolean negation or complementation. Indeed, $\mathcal{I} \models A_f \sqsubseteq \neg A_t$ in combination with the fact that $\mathcal{I} \not\models \varphi$ induces a partitioning of $\mathbb{D}_\mathcal{I}$, since, for $i \in [1,4]$, it holds that

$$\{c \in \mathbb{D}_\mathcal{I} \mid \mathcal{I}, \gamma[y_i := c] \not\models \forall z_i(\mathit{hasValue}(y_i, z_i) \wedge A_t(y_i))\} = \mathbb{D}_\mathcal{I} \setminus \{c \in \mathbb{D}_\mathcal{I} \mid \mathcal{I}, \gamma[y_i := c] \models \exists z_i(\mathit{hasValue}(y_i, z_i) \wedge \neg A_t(y_i))\}.$$

(Membership) We show that we we can (polynomially) reduce KBQA for (A)TCQs and Lite-English, COP+TV and COP knowledge bases to COKBSAT for $\mathbf{FO}^2$, i.e., to the unsatisfiability problem for $\mathbf{FO}^2$ knowledge bases. Let φ be an $\forall$-ATCQ and $(\mathcal{O}, \mathcal{D})$ be a Lite-English, COP or COP+TV knowledge base.

As we discussed before, we can reduce answering φ over knowledge base $(\mathcal{O}, \mathcal{D})$ to answering its core $\tilde{\varphi}$, which is an $\forall$-TCQ. Now, the meaning representations of Lite-English, COP+TV and COP are contained in $\mathbf{FO}^2$, whence $(\mathcal{O}, \mathcal{D})$ is a $\mathbf{FO}^2$ knowledge base. On the other hand, the $\forall$-TCQ $\tilde{\varphi}$ is a formula from $\mathbf{FO}^2$. Since $\mathbf{FO}^2$ is closed under negation, this means that $\neg\tilde{\varphi}$ is a also a $\mathbf{FO}^2$ formula. Notice that all these transformations are independent from the data and do not affect data complexity. Moreover,

$$(\mathcal{O}, \mathcal{D}) \models \tilde{\varphi} \quad \text{iff} \quad (\mathcal{O} \cup \{\neg\tilde{\varphi}\}, \mathcal{D}) \text{ is unsatisfiable.} \tag{‡}$$

where $(\mathcal{O} \cup \{\neg\tilde{\varphi}\}, \mathcal{D})$ is a $\mathbf{FO}^2$ knowledge base. This reduction is trivially sound and complete.

Since the data complexity of KBSAT for $\mathbf{C}^2$ and a fortiori for $\mathbf{FO}^2$ (that it subsumes) is in **NPTime** (see Chapter 3, Theorem 3.4.4), the result follows immediately. □

Theorem 5.5.5. *Answering $\neg$-TCQs over knowledge bases $(\mathcal{O}, \mathcal{D})$ where $\mathcal{O}$ is an empty ontology is* **coNPTime***-hard in data complexity. It is in* **coNPTime** *for Lite-English, COP+TV, COP and $\neg$-ATCQs.*

Proof. By reduction, again from 2+2-SAT. The proof is a variation of the previous one. We put $\mathcal{O} := \emptyset$, leave $\mathcal{D}_\varphi$ unchanged and consider the (boolean) $\neg$-TCQ

$$\begin{aligned}\psi \quad := \quad & \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (\mathit{pos}_1(x, y_1) \wedge \neg A_t(y_1) \wedge \mathit{pos}_2(x, y_2) \wedge \neg A_t(y_2) \wedge \\ & \mathit{neg}_1(x, y_3) \wedge A_t(y_3) \wedge \mathit{neg}_2(x, y_4) \wedge A_t(y_4))\end{aligned}$$

The intuition is that a propositional atom l of a 2+2 formula is true under some truth assignment $\delta(\cdot)$ iff $l \in A_t^\mathcal{I}$, for some $\mathcal{I} \models (\emptyset, \mathcal{D}_\psi)$, holds. The negation in φ induces again a partitioning

of $\mathbb{D}_{\mathcal{I}}$ into $A_t^{\mathcal{I}}$ and $\mathbb{D}_{\mathcal{I}} \setminus A_f^{\mathcal{I}}$ and allows us to "simulate" full boolean negation, thus ensuring the soundness and the completeness of the reduction.

For the upper bound we reason as in the previous theorem, by observing that the cores $\tilde{\varphi}$ of $\neg$-ATCQs φ are also contained in $\mathbf{FO}^2$, together with Lite-English, COP and COP+TV meaning representations. □

5.5.3 (¬)≤-(A)TCQs

Aggregation functions require data value domains. Moreover, real-world databases and hence, OBDAS, contain concrete data (integers, floats, strings, etc.). The question is: how difficult is (i.e., which is the data complexity of) query evaluation over OBDASs when we consider all the possible *orderings* to which a domain of interest may be subject? Orderings of domains can be partial, total, strict, dense, discrete, may or may have endpoints (a greatest or a least element).

Lemma 5.5.6. KBQA *for* $\leq$*-TCQs and empty ontologies* $\mathcal{O}$ *is* **coNPTime***-hard in data complexity.*

Proof. The proof is by reduction again from 2+2-SAT. Let $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ be a 2+2 formula over the propositional atoms $At(\psi) := \{l_1, \ldots, l_m\}$ with, for $i \in [1, k]$,

$$\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}.$$

Assume that $\mathbf{FO}$ frames $\mathbf{Dom}$ are totally ordered by some relation $\leq_{\mathbf{Dom}}$. The reduction proceeds as follows:

– We map formula ψ to the knowledge base $(\emptyset, \mathcal{D}_\psi)$, where $adom(\mathcal{D}_\psi)$ is totally ordered by $\leq_{\mathbf{Dom}}$:

$$\{pos_1(c_1, p_{11}), pos_2(c_1, p_{12}), neg_1(c_1, n_{11}), neg_2(c_1, n_{12}),$$
$$\vdots$$
$$pos_1(c_k, p_{k1}), pos_2(c_k, p_{k2}), neg_1(c_k, n_{k1}), neg_2(c_k, n_{k2}), Z(d)\},$$

– We consider the (boolean) $\leq$-TCQ

$$\varphi := \exists x \exists y_1 \exists y_2 \exists y_3 \exists y_4 (pos_1(x, y_1) \wedge \exists z_1(y_1 \leq z_1 \wedge Z(z_1)) \wedge pos_2(x, y_2) \wedge \exists z_2(y_2 \leq z_2 \wedge Z(z_2)) \wedge neg_1(x, y_3) \wedge \exists z_3(y_3 > z_3 \wedge Z(z_3)) \wedge neg_2(x, y_4) \wedge \exists z_4(y_4 > z_4 \wedge Z(z_4))).$$

Computing $(\mathcal{O}, \mathcal{D}_\psi)$ and φ is polynomial on the number of propositional atoms of ψ, which is $\leq 4k$. Furthermore, the only input of the of the reduction that depends on ψ is $\mathcal{D}_\psi$. Therefore, we are reasoning on the data complexity of KBQA. We now claim that

$$\psi \text{ is unsatisfiable} \quad \text{iff} \quad (\emptyset, \mathcal{D}_\psi) \models \varphi. \tag{†}$$

(⇒) Suppose that $(\emptyset, \mathcal{D}_\psi) \not\models \varphi$. We want to show that ψ is satisfiable. We know that $(\emptyset, \mathcal{D}_\psi) \not\models \varphi$ iff there exists an interpretation $\mathcal{I} \in Mod(\mathcal{D}_\psi)$ s.t. $\mathcal{I} \not\models \varphi$. Now, since $\mathcal{I} \models \mathcal{D}_\psi$, this means that $\mathcal{I}$ makes true all the atoms of $\mathcal{D}_\psi$, $pos_1(c_1, p_{11})$, $neg_1(c_1, n_{11})$, etc. On the other hand, $\mathcal{I} \not\models \varphi$, implies that, for all assignments $\gamma\colon \{x, y_1, \ldots, y_4, z_1, \ldots, z_4\} \to \mathbb{D}_{\mathcal{I}}$, whenever $\gamma(x) = c_i$, for $i \in [1, k]$, either *(i)* $\gamma(y_1) >_{\mathbf{Dom}} d$ and $\gamma(y_1) = p_{i1}$, or *(ii)* $\gamma(y_2) >_{\mathbf{Dom}} d$ and $\gamma(y_1) = p_{i2}$, or *(ii)* $\gamma(y_3) \leq_{\mathbf{Dom}} d$ and $\gamma(y_1) = n_{i1}$, or *(ii)* $\gamma(y_4) \leq_{\mathbf{Dom}} d$ and $\gamma(y_1) = n_{i2}$.

That is, every satisfying assignment γ over every model $\mathcal{I}$ of $(\emptyset, \mathcal{D}_\psi)$ "sets" the non-negated atoms of conjunct ψ_i to 1 and the negated atoms to 0. On the basis of this, we define a truth assignment $\delta\colon At(\psi) \to \{0, 1\}$ by putting, for $l \in At(\psi)$,

$$\delta(l) := \begin{cases} 1, & \text{if } \gamma(l) >_{\mathbf{Dom}} d, \\ 0, & \text{otherwise.} \end{cases}$$

Ordering	(¬)≤-(A)TCQs
TO (TOLG,TOL,TOG)	**coNPTime**-complete
TPO (TPOLG,TPOL,TPOG)	**coNPTime**-complete

Table 5.3: Data complexity of KBQA for (¬)≤-(A)TCQs, $\forall^*$-ontologies and total universal orderings.

Clearly, for all $i \in [1,k]$, it holds that $\delta(p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}) = 1$, and hence that $\delta(\psi) = 1$.

($\Leftarrow$) Assume that ψ is satisfiable. We build a database $\mathcal{D}$ such that $\mathcal{I}(\mathcal{D}) \models (\emptyset, \mathcal{D}_\psi)$ but $\mathcal{I}(\mathcal{D}) \not\models \varphi$. If ψ is satisfiable this means that there exists a truth assignment $\delta\colon \mathit{At}(\psi) \to \{0,1\}$ such that, for all $i \in [1,k]$, $\delta(\psi_i) = 1$. Database $\mathcal{D}$ can be therefore defined as the database $\mathcal{D} \supseteq \mathcal{D}_\psi$ where, for all $i \in [1,k]$ and $j \in \{1,2\}$,

$$\begin{gathered} \mathit{pos}_j(c_i, p_{ij}) \in \mathcal{D} \text{ when } \delta(p_{ij}) = 1, \\ \mathit{pos}_j(c_i, n_{ij}) \in \mathcal{D} \text{ when } \delta(n_{ij}) = 1, \\ p_{ij} \leq d \in \mathcal{D} \text{ when } \delta(p_{ij}) = 1 \text{ and} \\ Z(d) \in \mathcal{D}, \end{gathered}$$

whence $\mathcal{I}(\mathcal{D}) \models (\emptyset, \mathcal{D}_\psi)$, but $\mathcal{I}(\mathcal{D}) \not\models \varphi$.

Notice that the ordering induces a partitioning of the domain and is used to "simulate" boolean negation or complementation. □

To derive a **coNPTime** data complexity upper bound we consider different kinds of possible orderings of the domain of interest and ontologies for which a *finite* class of Herbrand models can be constructed. In particular, we will show that a non-deterministic polynomial time query answering algorithm exists for all the resulting combinations, with the exception dense orderings. We exploit the fact that the active domain $\mathit{adom}(\mathcal{D})$ of a knowledge base or OBDAS $(\mathcal{O}, \mathcal{D})$ is ordered provided that the models $\mathcal{I} \in \mathit{Mod}(\mathcal{O} \cup \mathcal{D})$ satisfy the different $\mathbf{FO}$ axiomatisations of orderings.

Definition 5.5.7. Let $\mathcal{I} = (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ be a a $\mathbf{FO}$ interpretation. A relation $S^\mathcal{I} \subseteq \mathbb{D}_\mathcal{I} \times \mathbb{D}_\mathcal{I}$ over $\mathcal{I}$'s domain is said to

– be a *strict partial ordering* (O) or, simply, an *ordering*, when $\mathcal{I}$ is a model of

$$\forall x \neg S(x,x) \textbf{ (irreflexivity)}$$
$$\forall x \forall y \forall z (S(x,y) \wedge S(y,z) \Rightarrow S(x,z)) \textbf{ (transitivity)}$$

– be *total* (T), when $\mathcal{I}$ is a model of

$$\forall x \forall y (S(x,y) \vee S(y,x) \vee x = y) \textbf{ (tricotomy)}$$

– have a *least element* (L) when $\mathcal{I}$ is a model of

$$\exists x \forall y S(x,y)$$

– have a *greatest element* (G) when $\mathcal{I}$ is a model of

$$\exists x \forall y S(y,x)$$

– be a *partial ordering* (PO) when $\mathcal{I}$ is a model of

$$\forall x S(x,x) \textbf{ (reflexivity)}$$
$$\forall x \forall y (S(x,y) \wedge S(y,x) \Rightarrow x = y) \textbf{ (antisymmetry)}$$
$$\forall x \forall y \forall z (S(x,y) \wedge S(y,z) \Rightarrow S(x,z)) \textbf{ (transitivity)}$$

These axioms can be combined together into: *total orderings* (TO), *total dense orderings* (TODen), *orderings with endpoints* (OLG), etc.

A $\mathbf{FO}$ *universal* or $\forall^*$-formula is a formula in prenex normal normal form $\varphi := \forall x_1 \cdots \forall x_n \psi$, where $\forall x_1 \cdots \forall x_n$ is a sequence of $n \geq 0$ universal quantifier prefixes and ψ is quantifier-free.

The following orderings are called *universal orderings* because their axioms are $\forall^*$-formulas: O, T, PO, TPO and TO. If we Skolemize such axiomatisations universal orderings can be extended to all the possible combinations of L and G with O, T, PO, TPO and TO. Notice that dense orderings, are not universal in this sense, since characterized by the $\forall^*\exists^*$-sentence (or axiom) $\forall x \forall y (S(x,y) \Rightarrow \exists z (S(x,y) \wedge S(z,y)))$.

A $\forall^*$*-ontology* $\mathcal{O}$ is a set of *DL-Lite*$_\sqcap$ inclusion assertions $A \sqsubseteq A'$, $\exists R \sqsubseteq A$, $A \sqsubseteq \neg A'$ and $\exists R \sqsubseteq \neg A$, all of which yield, when translated into $\mathbf{FO}$, (function-free) $\forall^*$-formulas (sentences, to be more precise). Notice that COP is included in this fragment of *DL-Lite*$_\sqcap$.

Lemma 5.5.8. KBQA *for* $\neg\leq$*-ATCQs, universal ontologies* $\mathcal{O}$ *and total universal orderings is in* **coNPTime** *w.r.t. data complexity.*

Proof. Let φ be a fixed $\neg\leq$-ATCQ, $(\mathcal{O}, \mathcal{D})$ a knowledge base where $\mathcal{O}$ is $\forall^*$-ontology, and assume that both the active domain $adom(\mathcal{D})$ of the database $\mathcal{D}$ and the domain $\mathbb{D}_\mathcal{I}$ of every model $\mathcal{I}$ of $(\mathcal{O}, \mathcal{D})$ are ordered by TO (TOLG,TOL,TOG) or TPO (TPOLG,TPOL,TPOG). Consider now a sequence $\bar{c}$ of constants. The equivalence

$$\bar{c} \notin cert(\tilde{\varphi}, \mathcal{O}, \mathcal{D}) \quad \text{iff} \quad \bar{c} \notin \bigcap \{ans(\tilde{\varphi}, \mathcal{D}') \mid \mathcal{D} \subseteq \mathcal{D}' \text{ and } \mathcal{I}(\mathcal{D}') \models \mathcal{O}\} \qquad (\dagger)$$

holds whenever $\bar{c} \notin ans(\tilde{\varphi}, \mathcal{D}')$ for some $\mathcal{D}'$ s.t. $\mathcal{I}(\mathcal{D}') \models (\mathcal{O}, \mathcal{D})$. It thus suffices to show that we can guess such a database and check whether $\bar{c} \notin ans(\tilde{\varphi}, \mathcal{D}')$ in time polynomial in $\#(adom(\mathcal{D}))$.

As $\mathcal{O}$ contains only universal sentences, the class $Mod(\mathcal{O} \cup \mathcal{D})$ of models of $(\mathcal{O}, \mathcal{D})$ coincides with the class $Mod_h(\mathcal{O} \cup \mathcal{D})$ of its Herbrand models. In particular, $Mod_h(\mathcal{O} \cup \mathcal{D})$ constitutes a finite lattice of finite (Herbrand) models ordered by the subinterpretation $\trianglelefteq$ ordering (see [Lal97], Proposition IV-5). Furthermore, for each $\mathcal{H} \in Mod_h(\mathcal{O} \cup \mathcal{D})$, $\mathbb{D}_\mathcal{H} \subseteq adom(\mathcal{D})$.

Construct the database $\mathcal{D}'$ as follows. Guess a $\mathcal{H} \in Mod_h(\mathcal{O} \cup \mathcal{D})$. We can guess $\mathcal{H}$ in time polynomial in $\#(adom(\mathcal{D}))$. Database $\mathcal{D}'$ will then be the database s.t. $\mathcal{H} = \mathcal{I}(\mathcal{D}')$ and can be computed from $\mathcal{H}$ in time polynomial in $\#(adom(\mathcal{D}))$.

Finally, to check whether $\bar{c} \notin ans(\tilde{\varphi}, \mathcal{D}')$, check whether there exists an assignment γ s.t. $\mathcal{H}, \gamma \not\models \tilde{\varphi}$, and $\gamma(\bar{x}) = \bar{c}$. Since φ is a $\mathbf{FO}$ formula this can be done in time polynomial in $\#(adom(\mathcal{D}))$ (see [AHV95], Theorem 17.1.1). Table 5.3 summarizes these results. □

Theorem 5.5.9. KBQA *is*

1. **coNPTime**-*complete in data complexity for* $\leq$*-(A)TCQs, universal COP ontologies and universal orderings.*
2. **coNPTime**-*hard in data complexity for* $\leq$*-(A)TCQs, Lite-English and COP+TV ontologies and total orderings.*

	$\vee$-(A)TCQs	$\leq$-(A)TCQs	$\forall$-(A)TCQs	$\neg$-(A)TCQs
Lite-English	in **LSpace**	**coNPTime**-hard	**coNPTime**-complete	**coNPTime**-complete
COP	in **LSpace**	**coNPTime**-complete	**coNPTime**-complete	**coNPTime**-complete
$\forall^*$-ontologies	in **LSpace**	**coNPTime**-complete	**coNPTime**-complete	**coNPTime**-complete
COP+TV	in **PTime**	**coNPTime**-hard	**coNPTime**-complete	**coNPTime**-complete

Table 5.4: Data complexity of aggregations w.r.t. $\vee$, $\forall$, $\neg$ and comparisons. We assume orderings to be *(i)* total and *(ii)* universal.

Proof. The first claim's upper bounds follow from Lemma 5.5.6. The first claim's lower bounds follow from Lemma 5.5.8. This is because one can transform by skolemization COP meaning representations into function-free $\forall^*$ sentences. Such transformation, which can be computed in time at most polynomial in the data, preserves (aggregate) certain answers. The second claim follows from Lemma 5.5.6. □

5.6 Summary

We have proposed a class of aggregate queries, viz., tree-shaped aggregate queries (ATCQs), equipped with a certain answers semantics. ATCQs can be considered a subclass of the so-called epistemic aggregate queries defined in [CNKT08]. ATCQs provide a $\mathbf{FO}$ formal notation for a significant number of SQL aggregate queries, and are built on top of GCQs and TCQs. Aggregate SQL queries are `SELECT-PROJECT-JOIN` queries with aggregate functions (`COUNT`, `AVG`, `SUM`, etc.), nested sub-queries, and `GROUP BY` and `HAVING` clauses.

We have shown how to express ATCQs in controlled English by means of the controlled language ATCQ-English, the extension of GCQ-English to aggregations. We analyze `GROUP BY` clauses as modifiers of the question's subject (i.e., its subject **N** constituent). By using higher ordering logic ($\mathbf{HO}$) and, hence, (bag) typed, intermediate semantic representations, we ensure that the translation $\tau(\cdot)$ is compositional. To express, in particular, SQL aggregate functions, which are defined over bags of values (i.e., groups), we propose a class of aggregate generalized determiners.

We also show that the $\mathbf{HO}$ meaning representations obtained are sound and complete w.r.t. the certain answers of the targeted ATCQs providing a further justification of *(i)* our particular notion of certain answers and of *(ii)* considering some kind of SQL query as the intended meaning of a question with sums, counts, superlatives and/or comparatives. We pinpoint some (possible) residual ambiguities that arise from the bag-set semantics of the ATCQs (and not of our controlled language questions which map into an unique ATCQ, up to structural equivalence).

We have also considered other ways of extending ATCQ- and GCQ-English by considering comparisons/comparatives, full negation and universal restrictions in queries. This done, we show (see Tables 5.3 and 5.4) that, whereas answering queries with aggregations can be (polynomially) reduced to answering those same queries without, and that, therefore, aggregate operations do not significantly increase data complexity, negations, universal restrictions and comparisons, yield intractability (by reduction from the **coNPTime**-complete 2+2-SAT problem):

- KBQA for ATCQs with disjunctions and either COP or Lite-English ontologies is in **LSpace** w.r.t. data complexity.
- KBQA for ATCQs with comparisons, negations and universal restrictions and disjointness assertions (expressible by COP, COP+TV and Lite-English) is **coNPTime**-hard w.r.t. data complexity.

Chapter 6

DL-English and the $\{\text{IS-A}_i\}_{i\in[0,7]}$ Family

Recently [BKGK05, MHWB06, Sch08, SLH03, SKC+08, ST06], controlled language interfaces to ontology-based data access systems (OBDASs) centered mostly around the W3C standard ontology language OWL[1] (Web Ontology Language) [FK06, KF07] have been proposed. They have given rise to a number of applications and implementations [BKGK05, MHWB06], among which ACE-OWL [FK06, KF07], which maps to OWL-DL and fragments of it (such as OWL-Lite). The formal underpinning of OWL-DL is provided by description logics [BCN+03, HPSv03], in particular, OWL Lite corresponds to the description logic $\mathcal{SHIF}$ (Actually $\mathcal{SHIF}(D)$, but we do not consider datatypes).

The data complexity of query answering in $\mathcal{SHIF}$, and thus in OWL Lite and ACE-OWL Lite (the fragment of ACE that maps to OWL Lite) is known to be **coNPTime**-complete: $\mathcal{SHIF}$ is subsumed by the description logic $\mathcal{ALCHQI}$ for which these computational properties hold (recall Theorem 3.3.5 from Chapter 4). Hence, controlled languages like ACE-OWL do not scale to data, although they contain fragments that do. This computational behavior depends on the language constructs they cover. It would be of interest, therefore, for controlled language designers working with OBDASs to know which natural language constructs (and in which combinations) give rise to this computational properties.

In Chapter 5, we have shown how to express the description logic *DL-Lite* and TCQs, for which KBQA is in **LSpace** [CdL+06], with the controlled languages Lite-English and GCQ-English, respectively. In this chapter we extend those results by considering fragments of ACE-OWL that are *(i) maximal* w.r.t. tractable data complexity (i.e., in **PTime**) when combined with TCQs and/or GCQ-English questions, and hence scale to data, and *(ii) minimal* w.r.t. intractable data complexity (i.e., **coNPTime**-hard), and hence do not. The results of this chapter will appear in [TC10a].

6.1 Expressing the Description Logic $\mathcal{ALCI}$ with DL-English

Figure 6.2 introduces thet grammar of the controlled language DL-English. Following the usual description logic conventions [BCN+03], we associate (and map) the non-recursive word categories $\mathbf{N}$, $\mathbf{Adj}$ and $\mathbf{IV}$ to atomic concepts. Category $\mathbf{TV}$ is associated to role names. Recursive constituents, by contrast, are associated to arbitrary concepts. For reasons of simplicity and space, we disregard morphology and polarity issues. We also omit specifying the (open) class of content words. Figure 6.1. Accordingly, examples of sentences in DL-English (we spell out the meaning

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

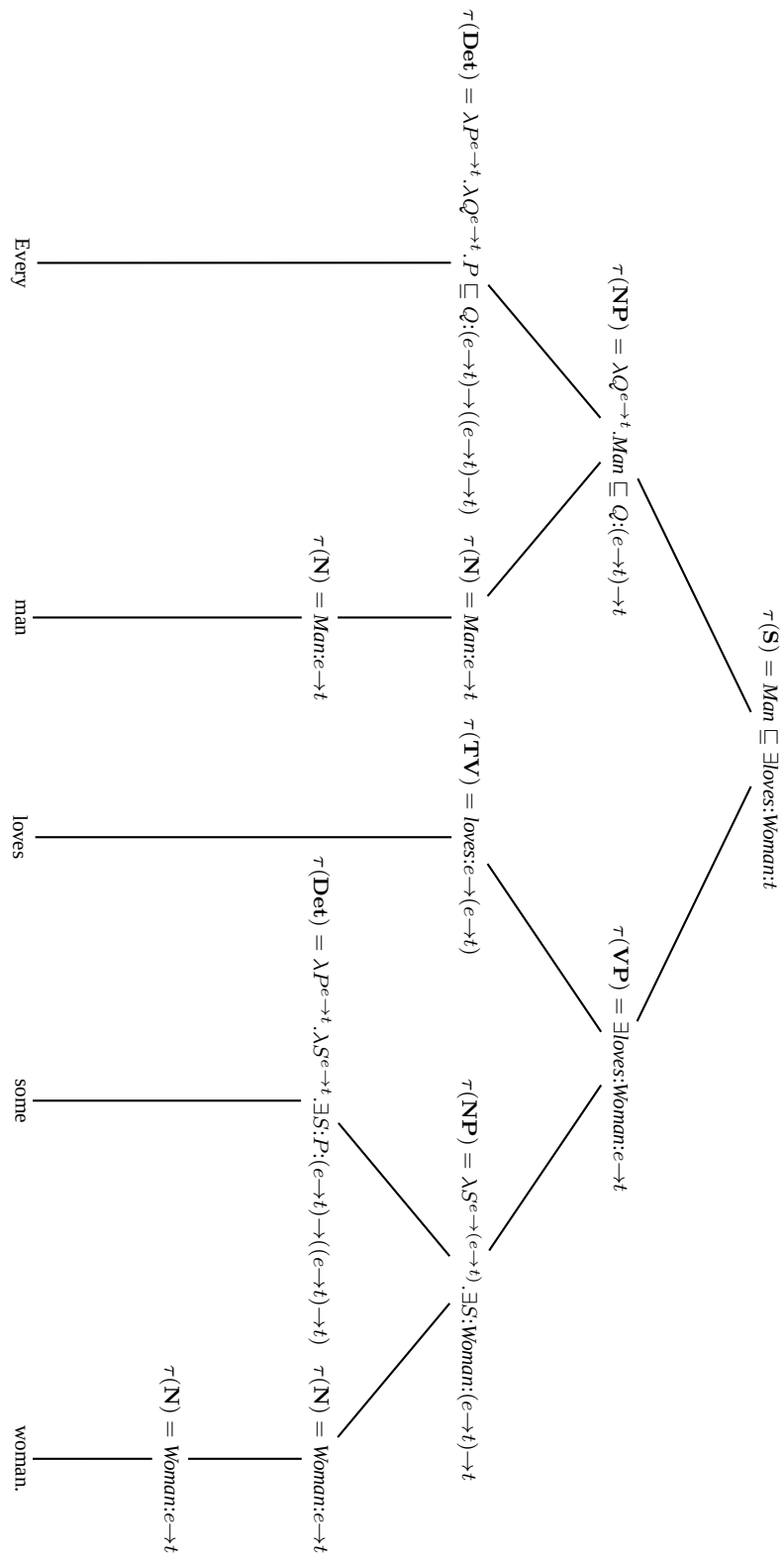


Figure 6.1: Parse tree for "Every man loves some woman.".

representations underneath) are:

$$\begin{array}{c}\text{No man who runs some business that does not make some money is shrewd.}\\ \textit{Man} \sqcap \exists \textit{run}{:}(\textit{Business} \sqcap \neg(\exists \textit{make}{:}\textit{Money})) \sqsubseteq \neg \textit{Shrewd}\end{array} \tag{6.1}$$

$$\begin{array}{c}\text{Nobody eats only apples.}\\ \forall \textit{eats}{:}\textit{Apple} \sqsubseteq \bot\end{array} \tag{6.2}$$

$$\begin{array}{c}\text{Everybody sleeps.}\\ \top \sqsubseteq \textit{Sleep}\end{array} \tag{6.3}$$

$$\begin{array}{c}\text{Every man loves some woman.}\\ \textit{Man} \sqcap \textit{Married} \sqsubseteq \exists \textit{has}{:}\textit{Wife}\end{array} \tag{6.4}$$

$$\begin{array}{c}\text{Anybody who has some car drives some new car or old car.}\\ \exists \textit{has}{:}\textit{Car} \sqsubseteq \exists \textit{drives}{:}((\textit{Car} \sqcap \textit{New}) \sqcup (\textit{Car} \sqcap \textit{Old}))\end{array} \tag{6.5}$$

We recall that derivations in context-free grammars and, hence, in semantically enriched grammars, are in one-to-one correspondance with parse trees and parsing, viz, the construction of such parse trees with the so-called history of the grammar derivation (see [JM09], Chapter 13).

More precisely, parsing amounts to walking or searching (depth or breadth first) a tree-shaped space of *parse states* $\delta := (w, \beta, T, E)$, where w stands for a controlled language syntactic constituent, β for its meaning representation (a $\mathbf{Ho}$ expression), T for its type and E for its typing context. Transition between states is based on unification and type-checking, and is fired by the grammar rules. Meaning representations are computed on the fly, during parsing. The states may also encode morphosyntactic information.

Such walk through the parsing space can be represented by means of (unification grammar-like) *derivation trees* whose nodes are the parse states δ and whose stand for transitions of two kinds: *(i)* OR-transitions, which point to the different possible successor states δ' of δ, and *(ii)* AND-transitions, which point to those states δ' to which the grammar rules applicable to δ may give rise. A derivation is said to be *succesful* if it results in a root state $(w, \beta, T, \emptyset)$ wherein w is a "well-typed" DL-English constituent (of semantics $\tau(w) = \beta{:}T$).

This notion of derivation tree and parse states will prove useful in showing why DL-English does not overgenerate, insofar as the typing of constituents prevent this from happening.

Lemma 6.1.1. *For all sentences D in DL-English, there exists an assertion α in $\mathcal{ALCI}$ s.t. $\tau(D) \equiv_s \alpha$.*

Proof. In order to prove this lemma we will prove something more general, namely, that,

$$\begin{array}{c}\text{for each } \mathbf{VP} \text{ or } \mathbf{N} \text{ constituent, there exists a concept } C \text{ in } \mathcal{ALCI}\\ \text{s.t. } \tau(\mathbf{VP}) \equiv_s C \text{ (resp. } \tau(\mathbf{VP}) \equiv_s C\text{).}\end{array} \tag{$\dagger$}$$

We prove ($\dagger$) by mutual induction in the length n (for $n \geq 1$) of DL-English derivations rooted in a $\mathbf{VP}$ or a $\mathbf{N}$. We make use of unification to prune away undesired parse (sub)trees when walking through the space of parsing states, whenever (semantic) types and (morphosyntactic) features fail to unify. See Figures 6.4 and 6.3 for examples.

– ($n = 1$). We consider in this case the derivations

$$\mathbf{VP} \Longrightarrow \mathbf{IV} \Longrightarrow A, \qquad \mathbf{VP} \Longrightarrow \text{is } \mathbf{Adj} \Longrightarrow A,$$
$$\mathbf{N} \Longrightarrow \text{is a } \mathbf{N} \Longrightarrow \text{is a } A,$$

which are in DL-English provided that we consider as part of our content lexicon the productions

$$\mathbf{IV} \to A, \tau(\mathbf{IV}) := A{:}e{\to}t, \qquad \mathbf{N} \to A, \tau(\mathbf{N}) := A{:}e{\to}t,$$
$$\mathbf{Adj} \to A, \tau(\mathbf{Adj}) := A{:}e{\to}t,$$

(Phrase structure rules)	(Semantic Actions)
$\mathbf{S} \to \mathbf{NP\ VP}$	$\tau(\mathbf{S}) := \tau(\mathbf{NP})(\tau(\mathbf{VP}))$
$\mathbf{NP} \to \mathbf{Det\ N}$	$\tau(\mathbf{NP}) := \tau(\mathbf{Det})(\tau(\mathbf{N}))$
$\mathbf{NP} \to \mathbf{Pro}$	$\tau(\mathbf{NP}) := \tau(\mathbf{Pro})$
$\mathbf{NP} \to \mathbf{Pro\ Relp\ VP}$	$\tau(\mathbf{NP}) := \tau(\mathbf{Pro})(\tau(\mathbf{Relp})(\tau(\mathbf{VP})))$
$\mathbf{VP} \to \mathbf{TV\ NP}$	$\tau(\mathbf{VP}) := \tau(\mathbf{NP})(\tau(\mathbf{TV}))$
$\mathbf{VP} \to$ is a $\mathbf{N}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{NP})(\tau(\mathbf{TV})))$
$\mathbf{VP} \to$ is $\mathbf{TV}$ by $\mathbf{NP}$	$\tau(\mathbf{VP}) := \tau(\mathbf{NP})(\tau(\mathbf{TV}))$
$\mathbf{VP} \to$ is $\mathbf{Neg\ Adj}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{Adj}))$
$\mathbf{VP} \to \mathbf{VP\ Crd\ VP}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Crd})(\tau(\mathbf{VP}))(\tau(\mathbf{VP}))$
$\mathbf{VP} \to$ is $\mathbf{Adj}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Adj})$
$\mathbf{VP} \to \mathbf{IV}$	$\tau(\mathbf{VP}) := \tau(\mathbf{IV})$
$\mathbf{VP} \to$ is $\mathbf{Neg\ TV}$ by $\mathbf{NP}$	$\tau(\mathbf{VP}) := \tau(\mathbf{NP})(\tau(\mathbf{Neg})(\tau(\mathbf{TV})))$
$\mathbf{VP} \to$ does $\mathbf{Neg\ IV}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{IV}))$
$\mathbf{VP} \to$ is $\mathbf{Neg}$ a $\mathbf{N}$	$\tau(\mathbf{VP}) := \tau(\mathbf{Neg})(\tau(\mathbf{N}))$
$\mathbf{N} \to \mathbf{N\ Relp\ VP}$	$\tau(\mathbf{N}) := \tau(\mathbf{N})(\tau(\mathbf{Relp})(\tau(\mathbf{VP})))$
$\mathbf{N} \to \mathbf{Adj\ N}$	$\tau(\mathbf{N}) := \tau(\mathbf{Adj})(\tau(\mathbf{N}))$
$\mathbf{N} \to \mathbf{N\ Crd\ N}$	$\tau(\mathbf{N}) := \tau(\mathbf{Crd})(\tau(\mathbf{N}))(\tau(\mathbf{N}))$

(Function lexicon)

$\mathbf{Pro} \to$ anybody	$\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q$
$\mathbf{Pro} \to$ somebody	$\tau(\mathbf{Pro}) := \lambda S^{e\to(e\to t)}.\exists S$
$\mathbf{Pro} \to$ nobody	$\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq \neg Q$
$\mathbf{Pro} \to$ nobody	$\tau(\mathbf{Pro}) := \lambda S^{e\to(e\to t)}.\neg\exists S$
$\mathbf{Crd} \to$ and	$\tau(\mathbf{Crd}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqcap Q$
$\mathbf{Crd} \to$ or	$\tau(\mathbf{Crd}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqcup Q$
$\mathbf{Relp} \to$ who	$\tau(\mathbf{Relp}) := \lambda P^{e\to t}.P$
$\mathbf{Relp} \to$ who	$\tau(\mathbf{Relp}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P : Q$
$\mathbf{Neg} \to$ not	$\tau(\mathbf{Neg}) := \lambda P^{e\to t}.\neg P$
$\mathbf{Pro} \to$ only	$\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\lambda S^{e\to(e\to t)}.\forall S{:}P$
$\mathbf{Pro} \to$ everybody	$\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\top \sqsubseteq P$
$\mathbf{Pro} \to$ nobody	$\tau(\mathbf{Pro}) := \lambda P^{e\to t}.P \sqsubseteq \bot$
$\mathbf{Det} \to$ some	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda S^{e\to(e\to t)}.\exists S{:}P$
$\mathbf{Det} \to$ every	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q$
$\mathbf{Det} \to$ no	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq \neg Q$

(Content Lexicon)

$\mathbf{TV} \to$ loves	$\tau(\mathbf{TV}) := \mathit{loves}{:}e\to(e\to t)$
$\mathbf{IV} \to$ sleeps	$\tau(\mathbf{IV}) := \mathit{sleeps}{:}e\to t$
$\mathbf{N} \to$ man	$\tau(\mathbf{N}) := \mathit{Man}{:}e\to t$
$\mathbf{Adj} \to$ new	$\tau(\mathbf{Adj}) := \lambda P^{e\to t}.P \sqcap \mathit{New}$
$\vdots$	$\vdots$

Figure 6.2: DL-English.

whence $\tau(\mathbf{TV}) = \tau(\mathbf{N}) \equiv_s A$, where A is an atomic concept.

– ($n = k + 1$). By induction hypothesis, for every derivation of length $i \leq k$ rooted in $\mathbf{VP}$ or $\mathbf{N}$, there exists a concept C s.t.

$$\mathbf{VP} \Longrightarrow^i w \text{ and } \tau(\mathbf{VP}) \equiv_s C \qquad \text{or} \qquad \mathbf{N} \Longrightarrow^i w \text{ and } \tau(\mathbf{N}) \equiv_s C, \tag{IH}$$

where w stands for a component derived (in DL-English) from $\mathbf{VP}$ (resp. $\mathbf{N}$) in $i \leq k$ steps. We want to prove that the property holds for $\mathbf{VP} \Longrightarrow^{k+1} w$ and $\mathbf{N} \Longrightarrow^{k+1} w$. We have several cases to consider, namely as many as there are recursive rules for $\mathbf{VP}$ and $\mathbf{N}$ in DL-English. In what follows, we will only look at some of them, given that the proof proceeds analogously for the remaining cases.

- $\mathbf{N} \Longrightarrow \mathbf{Adj}\,\mathbf{N} \Longrightarrow^k w\,w'$. By induction hypothesis, there exists a concept C' s.t. $\tau(w') \equiv_s C'$. Now, $\mathbf{Adj}$ is a *qualificative* adjective and we know from DL-English that in this case $\mathbf{Adj} \Longrightarrow A$ with meaning representation $\tau(\mathbf{Adj}) := \lambda D^{e\to t}.(A \sqcap D){:}(e\to t)\to(e\to t)$. Thus,

$$\begin{array}{rcl} \tau(A\,w') & =_{df} & \lambda P^{e\to t}.A \sqcap P(\tau(w')){:}e\to t \\ & =_{ih} & \lambda D^{e\to t}.A \sqcap P(C'){:}e\to t \\ & \rhd & A \sqcap C'{:}e\to t, \end{array}$$

  and $A \sqcap C'$ is the concept we were looking for. Notice that any other choice for $\mathbf{Adj}$ in the same position (i.e., with meaning representation $\tau(\mathbf{Adj}) := A{:}e\to t$) would have resulted in a non-derivation, due to $\mathbf{Ho}$ type constraints.

- $\mathbf{N} \Longrightarrow \mathbf{N}\,\mathbf{Relp}\,\mathbf{TV} \Longrightarrow^k w$ who w''. By induction hypothesis, there exists a concept C s.t. $\tau(w) \equiv_s C$ and a concept C'' s.t. $\tau(w'') \equiv_s C''$. On the other hand, we have that in DL-English $\tau(\mathbf{Relp}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.(P \sqcap Q){:}(e\to t)\to((e\to t)\to(e\to t))$. Hence,

$$\begin{array}{rcl} \tau(w \text{ who } w') & =_{df} & \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqcap Q(\tau(w))(\tau(w)){:}\ e\to t \\ & =_{ih} & \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqcap Q(C)(C''){:}e\to t \\ & \rhd & C \sqcap C''{:}e\to t, \end{array}$$

  and therefore, $\tau(\mathbf{VP}) \equiv_s C \sqcap C''$, which is an $\mathcal{ALCI}$ concept.

- $\mathbf{VP} \Longrightarrow \mathbf{TV}\,\mathbf{NP} \Longrightarrow \mathbf{TV}\,\mathbf{Det}\,\mathbf{N} \Longrightarrow^{k-1} ww'w''$. By induction hypothesis, there exists a concept C'' s.t. $\tau(w'') \equiv_s C''$. We know that in DL-English $\mathbf{TV} \Longrightarrow r$, with $\tau(\mathbf{TV}) \equiv_s r$. There are only two possibilities for $\mathbf{Det}$:

$$\mathbf{Det} \Longrightarrow \text{only} \qquad \text{or} \qquad \mathbf{Det} \Longrightarrow \text{some}.$$

  Let us focus, w.l.o.g. on the former. We know that in such case it holds that $\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda S^{e\to(e\to t)}.(\forall S{:}P){:}(e\to(e\to t))\to((e\to t)\to(e\to t))$. Therefore,

$$\begin{array}{rcl} \tau(r \text{ only } w'') & =_{df} & \lambda P^{e\to t}.\lambda S^{e\to(e\to t)}.\forall S{:}P(\tau(w''))(r){:}e\to t \\ & =_{ih} & \lambda P^{e\to t}.\lambda S^{e\to(e\to t)}.\forall S{:}P(C'')(r){:}e\to t \\ & \rhd & \forall r{:}C''{:}e\to t, \end{array}$$

  and, clearly, $\tau(\mathbf{VP}) \equiv_s \forall r{:}C''$. Notice that any other choice for $\mathbf{Det}$ would prevent any derivation of the whole constituent. For instance, while $\mathbf{Det} \Longrightarrow$ every, constituent of (partial) meaning representation $\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q{:}t$, we cannot apply $\lambda Q^{e\to t}.C'' \sqsubseteq Q{:}e\to t$ to $r\colon e\to(e\to t)$, due to our $\mathbf{Ho}$ type system.

- $\mathbf{VP} \Longrightarrow$ is $\mathbf{Neg}$ a $\mathbf{N} \Longrightarrow^k ww'$. By induction hypothesis, there exists a concept C' s.t. $\tau(w') \equiv_s C'$. Now, in DL-English we have that $\mathbf{N} \Longrightarrow$ not with meaning representation $\tau(\mathbf{Neg}) := \lambda P.\neg P{:}e{\to}t$. Therefore,

$$\begin{array}{rcl} \tau(\text{is not a } w') & =_{df} & \lambda P^{e\to t}.\neg P(\tau(w')){:}e{\to}t \\ & =_{ih} & \lambda P^{e\to t}.\neg P(C'){:}e{\to}t \\ & \rhd & \neg C'{:}e{\to}t, \end{array}$$

  and thus $\neg C'$ is the concept we were looking for.
- All the other cases are dealt with analogously.

We now turn to complete utterances, viz. to DL-English sentences. There a few number of different ways in which a sentence D can be generated in our controlled language, namely:

– $\mathbf{S} \Longrightarrow \mathbf{NP\,VP} \Longrightarrow \mathbf{Det\,N\,VP} \Longrightarrow^* ww'w''$ (with $S = ww'w''$). We know that $\tau(w') \equiv_s C'$ and that $\tau(w'') \equiv_s C''$. Due to the typing constraints, the only possibilities for **Det** are:

$$\mathbf{Det} \Longrightarrow \text{every} \qquad \text{or} \qquad \mathbf{Det} \Longrightarrow \text{no},$$

with meaning representation $\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q{:}(e{\to}t){\to}((e{\to}t){\to}t)$ (resp. $\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq \neg Q{:}(e{\to}t){\to}((e{\to}t){\to}t)$). Hence,

$$\begin{array}{rcl} \tau(\text{every } w'w'') & =_{df} & \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q(\tau(w'))(\tau(w'')) : t \\ & =_{\dagger} & \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq Q(C')(C'') : t \\ & \rhd & C' \sqsubseteq C'' : t. \end{array}$$

– $\mathbf{S} \Longrightarrow \mathbf{NP\,VP} \Longrightarrow \mathbf{Pro\,Relp\,VP\,VP} \Longrightarrow^* ww'w''w'''$ (with $S = ww'w''w'''$). Observe that the only two possible derivations rooted in **Pro** that yield a successful overall derivation are

$$\mathbf{Pro} \Longrightarrow \text{everybody} \qquad \text{or} \qquad \mathbf{Pro} \Longrightarrow \text{nobody},$$

with meaning representation, for the latter, $\tau(\mathbf{S}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.P \sqsubseteq \neg Q{:}(e{\to}t){\to}((e{\to}t){\to}t)$ (the former is similar). Similarly, the only partial derivation for $\mathbf{Relp}$ is

$$\mathbf{Relp} \Longrightarrow \text{who}$$

with meaning representation $\tau(\mathbf{Relp}) := \lambda P^{e\to t}.P{:}(e{\to}t){\to}(e{\to}t)$. By applying the same argument as before, together with the definition of $\tau(\cdot)$ given by DL-English, it follows that $\tau(\mathbf{S}) = C'' \sqsubseteq C''' : t$.

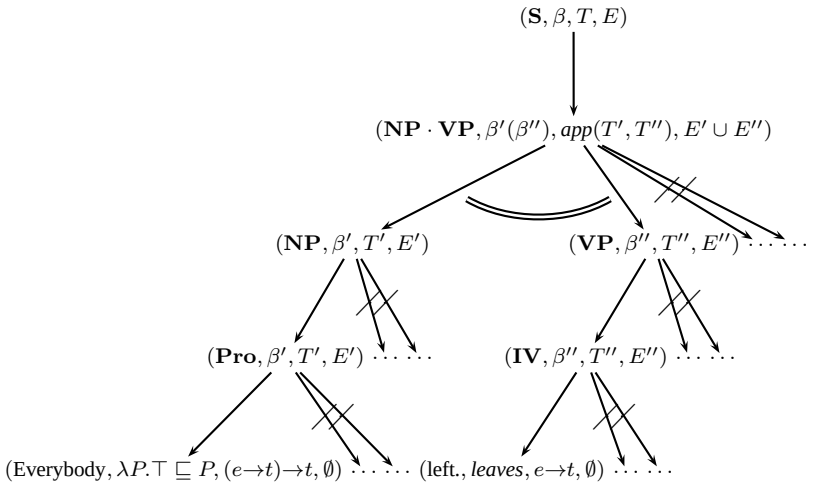
– $\mathbf{S} \Longrightarrow \mathbf{NP\,VP} \Longrightarrow \mathbf{Pro\,VP} \Longrightarrow^* ww'$ (with $\mathbf{S} = ww'$). The only two possibilities for $\mathbf{Pro}$ are the following:

$$\mathbf{Pro} \Longrightarrow \text{nobody} \qquad \text{or} \qquad \mathbf{Pro} \Longrightarrow \text{anybody}.$$

with meaning representation, resp. $\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\bot \sqsubseteq P : (e{\to}t){\to}t$ and $\tau(\mathbf{Pro}) := \lambda P^{e\to t}.\top \sqsubseteq P : (e{\to}t){\to}t$. Therefore, $\tau(\mathbf{S}) = \bot \sqsubseteq C' : t$ (resp., $\tau(\mathbf{S}) = \top \sqsubseteq C'{:}t$).

Therefore, for each D in DL-English there exists an assertion α s.t. $\tau(D) \equiv_s \alpha$. This closes the proof. □

Lemma 6.1.2. *For all assertions α in $\mathcal{ALCI}$, there exists a sentence D in DL-English s.t. (i) $\tau(D) \equiv_s \alpha'$, where α' is an $\mathcal{ALCI}$ assertion, and (ii) α' is equivalent to α.*



$$\frac{\vdash \lambda P^{e\to t}.\top \sqsubseteq P{:}(e{\to}t){\to}t \qquad \vdash \mathit{Leaves}{:}e{\to}t}{\vdash \lambda P^{e\to t}.\top \sqsubseteq P(\mathit{Leaves}){:}t}$$

$$\lambda P^{e\to t}.\top \sqsubseteq P(\mathit{Leaves}) \rhd \top \sqsubseteq \mathit{Leaves}$$

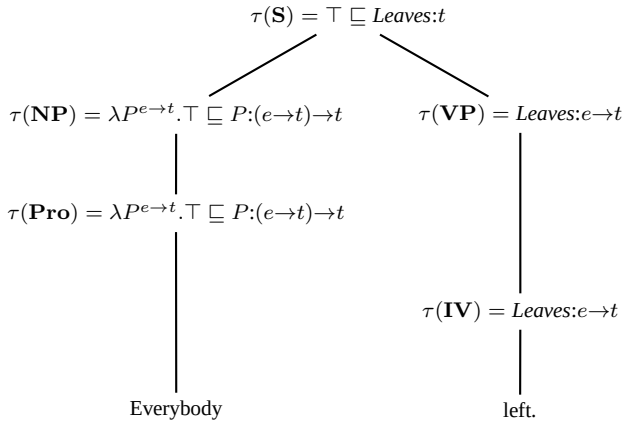


Figure 6.3: Top: A succesful derivation for "Everybody left.". The dots indicate failed transitions, $app(\cdot,\cdot)$ indicates a type unification function and angles indicate AND-transitions. **Middle:** Transitions only succeed when meaning can be applied to each other, their contexts merged and their types unified. **Bottom:** Resulting DL-English parse tree. The information propagated from leaves to root via unification yields, ultimately, the state (everybody left, $\top \sqsubseteq \mathit{leaves}, t, \emptyset$).

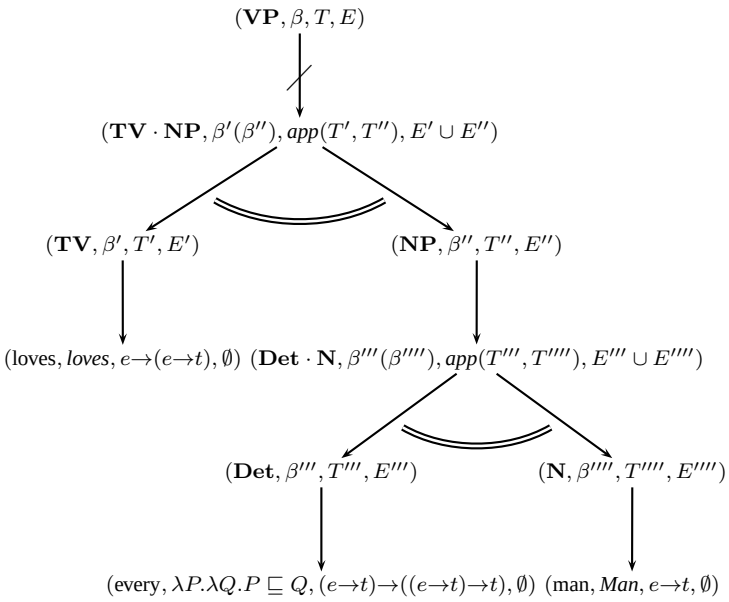


Figure 6.4: A failed derivation for the **VP** "loves every man", since $app(e{\to}(e{\to}t), e{\to}t)$ is undefined (unification is not possible). The string is not well-typed and is thus devoid of a meaning representation and a parse tree.

Proof. Again, in order to prove this lemma, we prove a more general claim, namely, that

$$\begin{array}{c}\text{for each concept } C \text{ in NNF, there exists either a } \mathbf{VP} \text{ or a } \mathbf{N}\\ \text{s.t. } \tau(\mathbf{VP}) \equiv_s C' \text{ or } \tau(\mathbf{N}) \equiv_s C',\end{array} \qquad (\dagger)$$

where C' is equivalent to C. This we prove by induction on C.

– (Basis). There are two cases to consider, given that C is in NNF.
  - $C := A$. Notice that A is already in NNF. Include in the function lexicon of DL-English the (terminal) production $\mathbf{N} \to A$ with lexical semantics $\tau(\mathbf{N}) := A{:}e{\to}t$. There are two possibilities:
    i. either $\mathbf{N} \Longrightarrow \mathbf{N} \Longrightarrow A$,
    ii. or $\mathbf{VP} \Longrightarrow$ is a $\mathbf{N} \Longrightarrow$ is a $\mathbf{N} \Longrightarrow$ is a A

    Notice, furthermore that we can express $\top$ with the ad hoc $\mathbf{N}$ "thing", which is the usual description logic convention [BCN+03].
  - $C := \neg A$. then $\mathbf{VP} \Longrightarrow$ is $\mathbf{Neg}$ a $\mathbf{N} \Longrightarrow$ is not a $\mathbf{N} \Longrightarrow$ is not a A, with $\tau(\mathbf{VP}) \equiv_s \neg A$, by the same argument as before, which is in NNF.

– (Inductive step). By inductive hypothesis we know that, for all subconcepts C' of C, there exists a component w rooted in a $\mathbf{VP}$ or $\mathbf{N}$ s.t.:

$$\mathbf{VP} \Longrightarrow^* w \text{ and } \tau(\mathbf{VP}) \equiv_s C'' \quad \text{or} \quad \mathbf{N} \Longrightarrow^* w \text{ and } \tau(\mathbf{N}) \equiv_s C'', \qquad \text{(IH)}$$

where C'' is a concept equivalent to C'. This leaves two cases to consider, namely:

  - $C := \exists r{:}C'$. By (IH), there exists a w' s.t. either $\mathbf{VP} \Longrightarrow^* w'$ and $\tau(\mathbf{VP}) \equiv_s C''$, or $\mathbf{N} \Longrightarrow^* w'$ $\tau(\mathbf{N}) \equiv_s C''$, and C'' is equivalent to C'. This gives us two possibilities for $\exists r{:}C$, namely:
    i. $\mathbf{VP} \Longrightarrow \mathbf{TV\,NP} \Longrightarrow \mathbf{TV\,Det\,N} \Longrightarrow^* r$ some w', with $sem\mathbf{VP}, \equiv_s \exists r{:}C'$, which is equivalent to $\exists r{:}C$, or

ii. $\mathbf{VP} \Longrightarrow \mathbf{TV\,NP} \Longrightarrow \mathbf{TV\,Pro\,Relp\,VP} \Longrightarrow^* r$ somebody who w', where $\tau(\mathbf{VP}), \equiv_s \exists r{:}C'$, which is equivalent to $\exists r{:}C$.

- $C := C' \sqcap C''$. Again, by (IH), there exists a w' (resp. a w'') s.t. either $\mathbf{VP} \Longrightarrow^* w'$ and $\tau(\mathbf{VP}) \equiv_s C'''$ (resp. C''''), or $\mathbf{N} \Longrightarrow^* w'\ \tau(\mathbf{N}) \equiv_s C'''$ (resp. C''''), and C''' is equivalent to C' (resp. C'''' is equivalent to C''). As before, we have two cases:
  i. $\mathbf{VP} \Longrightarrow \mathbf{VP\,Crd\,VP} \Longrightarrow^* w'$ and w'', whence $\tau(\mathbf{VP}) \equiv_s C''' \sqcap C''''$ and $C''' \sqcap C''''$ is equivalent to $C''' \sqcap C''$.
  ii. $\mathbf{N} \Longrightarrow \mathbf{N\,Relp\,VP} \Longrightarrow^* w'$ and w'', whence $\tau(\mathbf{N}) \equiv_s C'''\sqcap C''''$ and $C'''\sqcap C''''$ is equivalent to $C''' \sqcap C''$.

Let now be $C \sqsubseteq C'$ be an $\mathcal{ALCI}$ assertion with C, C' in NNF. We can capture this assertion in either of two ways in DL-English:

– either $\mathbf{S} \Longrightarrow^*$ every $\mathbf{N\,VP} \Longrightarrow^*$ every ww',
– or $\mathbf{S} \Longrightarrow^*$ everybody who $\mathbf{VP\,VP} \Longrightarrow^*$ everybody who ww',

for some (two) components w' and w' whose existence is guaranteed by (†). Clearly, in both cases $\tau(\mathbf{S}) \equiv_s C'' \sqsubseteq C'''$. Moreover, it is evident that $C'' \sqsubseteq C'''$ is equivalent to $C \sqsubseteq C'$. This closes the proof. □

From Lemmas 6.1.1 and 6.1.2 follows immediately that DL-English expresses $\mathcal{ALCI}$ (up to equivalence).

Theorem 6.1.3 (DL-English)**.** *DL-English expresses $\mathcal{ALCI}$.*

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

We now turn to the computational properties of each of the constructs *in isolation* of DL-English. We do it by essentially restricting the kind of *right* (i.e., C_r) and *left* (i.e., C_l) concepts we may express. All utterances comply with the sentence patterns

"every $w_l\ w_r$" and "everybody who $w_l\ w_r$".

The constituents w_l and w_r map to, respectively, left and right concepts, while sentences map to IS-A assertions of the form $C_l \sqsubseteq C_r$. We consider in this paper only 8 out of all possible combinations obtained by allowing in C_l and C_r some subset of the description logic constructs in Figure 6.2, giving rise to the family $\{\text{IS-A}_i\}_{i\in[0,7]}$ of controlled languages shown in Figure 6.1. The basic kind of assertion they all express is IS-A among atomic concepts, viz., $A \sqsubseteq A'$, captured by IS-A_0.

Theorem 6.2.1 (IS-A_is)**.** *For each $i \in [0,7]$ and each sentence D_i in IS-A$_i$, there exists an assertion α_i s.t. $\tau(D_i) \equiv_s \alpha_i$. Conversely, for each assertion α_i there exists a sentence D_i in IS-A$_i$ s.t. $\tau(D_i) \equiv_s \alpha_i'$ and α_i' is equivalent to α_i.*

Proof. The theorem can be proved for each fragment in a manner analogous to DL-English. Basically, we consider two cases both on the ($\Rightarrow$) and the ($\Leftarrow$) directions of the proof, viz., a (mutual) induction on either *(i)* $\mathbf{N}_l$ and $\mathbf{VP}_l$ constituents or *(ii)* on the $\mathbf{N}_r$ and $\mathbf{VP}_r$ constituents for the if direction and a (mutual) induction over *(i)* a C_l or *(ii)* a C_r concept for the only if direction. With some routine adjustments to the specific syntax of the fragments and their meaning representations, we adapt each time the proof of Theorem 6.1.3. □

$\mathbf{S} \to \mathbf{NP}_l\,\mathbf{VP}_r \quad \mathbf{NP}_l \to \mathbf{Pro}_l\,\mathbf{Relp}_l\,\mathbf{VP}_l \quad \mathbf{NP}_l \to \mathbf{Det}_l\mathbf{N}_l$
$\mathbf{Pro}_l \to$ anybody $\quad \mathbf{Relp}_l \to$ who $\quad \mathbf{Det}_l \to$ every

Concept C_f	**Constituent** γ_f	**Grammar Rules**
$\exists r{:}A$	**TV** some $\mathbf{N}_f$ **TV** somebody who $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** $\mathbf{N}_f \to \mathbf{N}$
$\exists r^{-}{:}A$	is **TV** by some $\mathbf{N}_f$ is **TV** by somebody who $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** | **TV** $\mathbf{NP}_f$ $\mathbf{N}_f \to \mathbf{N}$ $\mathbf{NP}_f \to \mathbf{Det}_f\,\mathbf{N}_f$ | $\mathbf{Pro}_f\,\mathbf{Relp}_f\,\mathbf{VP}_f$ $\mathbf{Det}_f \to$ some $\mathbf{Relp}_f \to$ who $\mathbf{Pro}_f \to$ somebody
$\forall r{:}A$	**TV** only $\mathbf{VP}_f$ **TV** only who $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** | **TV** $\mathbf{NP}_f$ $\mathbf{N}_f \to \mathbf{N}$ $\mathbf{NP}_f \to \mathbf{Det}_f\,\mathbf{N}_f$ | $\mathbf{Pro}_f\,\mathbf{Relp}_f\,\mathbf{VP}_f$ $\mathbf{Det}_f \to$ only $\mathbf{Relp}_f \to$ who $\mathbf{Pro}_f \to$ only
A	$\mathbf{N}_f$ $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** $\mathbf{N}_f \to \mathbf{N}$
$A_1 \sqcap \cdots \sqcap A_n$	$\mathbf{Adj}\,\mathbf{N}_f$ $\mathbf{N}_f$ who $\mathbf{VP}_f$ $\mathbf{N}_f$ and $\mathbf{N}_f$ $\mathbf{VP}_f$ and $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** | $\mathbf{VP}_f\,\mathbf{Crd}_f\,\mathbf{VP}_f$ $\mathbf{N}_f \to \mathbf{N}$ | $\mathbf{Adj}\,\mathbf{N}_f$ | $\mathbf{N}_f\,\mathbf{Crd}_f\,\mathbf{N}_f$ | $\mathbf{N}_f\mathbf{Relp}_f\,\mathbf{VP}_f$ $\mathbf{Relp}_f \to$ who $\mathbf{Crd}_f \to$ and
$\exists r$	**TV** something **TV** somebody	$\mathbf{VP}_f \to \mathbf{TV}\,\mathbf{Pro}_f$ $\mathbf{Pro}_f \to$ somebody | something
$A_1 \sqcup \cdots \sqcup A_n$	$\mathbf{VP}_f$ or $\mathbf{VP}_f$	$\mathbf{VP}_f \to$ is a $\mathbf{N}_f$ | **IV** | is **Adj** | $\mathbf{VP}_f$ and $\mathbf{VP}_f$ $\mathbf{N}_f \to \mathbf{N}$ | $\mathbf{N}_f$ and $\mathbf{N}_f$
$\neg A$	is not **Adj** does not **IV** is not a $\mathbf{N}_f$	$\mathbf{N}_f \to \mathbf{N}$ $\mathbf{VP}_f \to$ does not **IV** | is not **Adj** | is not a $\mathbf{N}_f$

Table 6.1: Expressing concepts C_f, for $f \in \{l, r\}$, and assertions $C_l \sqsubseteq C_r$, by restricting and subcategorizing rules in DL-English.

	Assertions α_i	**Example(s)**
IS-A_0	$A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	Every businessman is a cunning man
IS-A_1	$A \sqsubseteq \forall r{:}A$	Every herbivore eats only herbs eats only herbs.
IS-A_2	$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq \forall r{:}(A_1 \sqcap \cdots \sqcap A_k)$	Every Italian man drinks only strong coffee.
IS-A_3	$\exists r{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$ $\exists r^{-}{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$ $A \sqsubseteq \exists r$	Anybody who murders some person is a heartless killer. Anybody who is loved by some person is a happy person. Every driver drives something.
IS-A_4	$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq A_1 \sqcap \cdots \sqcap A_k$ $\exists r{:}(A_1 \sqcap \cdots \sqcap A_n) \sqsubseteq A_1 \sqcap \cdots \sqcap A_k$	Every cruel man is a bad man. Anybody who runs some bankrupt company is a bad businessman.
IS-A_5	$\forall r{:}A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	Anybody who values only money is a greedy person.
IS-A_6	$A \sqsubseteq A_1 \sqcup \cdots \sqcup A_n$	Every mammal is male or is female.
IS-A_7	$\neg A \sqsubseteq A_1 \sqcap \cdots \sqcap A_n$	Anybody who is not selfish is a reasonable person.

Table 6.2: Defining the $\{\text{IS-A}_i\}_{i\in[0,7]}$ controlled languages. Each IS-A_i, for $i > 0$, contains the assertions of IS-A_0.

6.3 Data Complexity

In this section we state the main data complexity results for the $\{\text{IS-A}_i\}_{i\in[0,7]}$ fragments.

Theorem 6.3.1. *The data complexity of* KBQA *for (T)CQs is*

1. *in* **LSpace** *for IS-A_0,*
2. **PTime***-complete for IS-A_2, IS-A_3 and IS-A_4 and*
3. **coNPTime***-complete for IS-A_5, IS-A_6, and IS-A_7.*

Proof. The controlled language IS-A_0 is subsumed by the controlled language Lite-English, which as we have shown elsewhere (see Theorem 4.1.4, Chapter 5) expresses the description logic *DL-Lite*$_\sqcap$ for which KBQA w.r.t. CQs is in **LSpace** in data complexity (see Theorem 3.3.5, Chapter 3).

The lower bounds for IS-A_2, IS-A_3 and IS-A_4 follow from the results in [CdL$^+$06]. 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 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. The complexity upper bounds, on the other hand, follow from results by [LK07] for the description logic $\mathcal{EL}$, which subsumes the description logic assertions IS-A_2, IS-A_3 and IS-A_4 express and hold for CQs.

The lower bounds for IS-A_5, IS-A_6, and IS-A_7 follow also from [CdL$^+$06]: 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. In these three cases, TCQs are used to define a reduction from the **NPTime**-complete 2+2-SAT problem (recall Chaper 6, Section 5.5). The **coNPTime** upper bounds for these fragments, on the

other hand, derive from the **coNPTime** data complexity upper bounds for KBQA over expressive description logics (containing $\mathcal{ALCI}$) shown in [OCE08] and hold, again, for CQs. □

Theorem 6.3.2. KBQA *for IS-A$_1$ and (T)CQs is* **NLSpace***-complete w.r.t. data complexity.*

Proof. (**Hardness**) Calvanese et al. show in [CdL$^+$06] (by reduction from the reachability problem for directed graphs) that any description logic capable of expressing assertions of the form $A \sqsubseteq \forall r.A'$, or, equivalently, of the form $\exists r^-.A \sqsubseteq A'$, is **NLSpace**-hard for KBQA. 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", rather than by sentences like "Every A rs every A'".

(**Membership**) Let $\varphi := \psi(\bar{x}, \bar{y})$ 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 will reduce KBQA for IS-A$_1$ to KBQA for linear **DATALOG**, which is known to be **NLSpace**-complete in data complexity (see [EGDV01], Theorem 4.3). The only inclusion assertions expressible by our fragment are $A \sqsubseteq A'$ and $\exists r.A \sqsubseteq A'$, which can be transformed into an (equivalent) set $\mathcal{P}_\mathcal{O}$ of clauses $\neg A(x) \vee A'(x)$ and $\neg r(x, y) \vee \neg A(x) \vee A'(y)$, called a linear **DATALOG** *program*. While CQ φ may not be a linear **DATALOG** goal, $\psi(\bar{x}, \bar{y})$ consists of a conjunction of k atoms $S_1(\bar{z}_1) \wedge \cdots \wedge S_k(\bar{z}_k)$, where $\bar{x} \cup \bar{y} = \bar{z}_1 \cup \cdots \cup \bar{z}_k$. If we were to transform such atoms into a family of atomic queries (which are linear **DATALOG** goals), by means of some satisfaction-preserving reduction that requires only $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ space, the data complexity upper bound would immediately follow.

Start by computing the program $\mathcal{P}_\mathcal{O}$ as described above. Since $\mathcal{O}$ is fixed, transforming it into $\mathcal{P}_\mathcal{O}$ does not affect data complexity. We transform now $\psi(\bar{x}, \bar{y})$, in space logarithmic in $\#(\mathcal{D})$, into such family of linear **DATALOG** goals, thus reducing answering φ over $\mathcal{O}$ and $\mathcal{D}$ to answering a family of atomic goals over $\mathcal{P}_\mathcal{O}$ and $\mathcal{D}$. Ground φ by $\sigma := \{\bar{x} \mapsto \bar{c}\}$. Grounding φ by σ, which returns (the CQ) $\psi(\bar{c}, \bar{y})$, does not affect, once again, data complexity. Next, consider all the possible groundings $\{\bar{y} \mapsto \bar{c}'\}$ with $\bar{c}' \in \mathit{adom}(\mathcal{D})^{|\bar{y}|}$ and apply them to $\psi(\bar{c}, \bar{y})$. There are $\mathbf{O}(\#(\mathcal{D})^{|\bar{y}|})$ such groundings. This yields a family of CQs $\psi(\bar{c}, \bar{c}')$, whose atoms can be stored in a registry of $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ size (we can encode such grounded atoms using $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ bits). This reduction is sound and complete. Indeed

$$(\mathcal{O}, \mathcal{D}) \models \psi(\bar{c}, \bar{y}) \quad \text{iff} \quad \begin{array}{l} \mathcal{P}_\mathcal{O} \cup \mathcal{D} \models S_i(\bar{c}''), \text{ for all } i \in [1, k] \text{ and} \\ \quad \text{some } \bar{c}'' \in \mathit{adom}(\mathcal{D})^{|\bar{z}_i|} \text{ "compatible" with } \bar{c}, \end{array} \tag{$\dagger$}$$

where by "compatible" we mean that $\bar{c}''$ coincides with $\bar{c}$ on the distinguished variables (note that $\bar{z}_i$ may contain *both* distinguished and non-distinguished variables).

The ($\Rightarrow$) direction is immediate. To prove the ($\Leftarrow$) 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 $\mathcal{I} \not\models S_i(\bar{c}'')$, for some $i \in [1, k]$ and every $\bar{c}'' \in \mathit{adom}(\mathcal{D})^{|\bar{z}_i|}$ "compatible", again, with $\bar{c}$. Since $\mathcal{I} \models \mathcal{P}_\mathcal{O} \cup \mathcal{D}$, we have that $\mathcal{I} \models (\mathcal{O}, \mathcal{D})$ and $\mathcal{I} \models \psi(\bar{c}, \bar{y})$. Therefore, for some grounding σ' from $\bar{y}$ into $\mathit{adom}(\mathcal{D})$, $\mathcal{I} \models S + +_i(\bar{z}_i)\sigma\sigma'$. Now, clearly, $\bar{c}'' = \bar{c}''_1 \cup \bar{c}''_2$ with $\bar{c}''_2 \subseteq \bar{c}$, and $\bar{z}_i = \bar{z}_{i1} \cup \bar{z}_{i2}$ with $\bar{z}_{i2} \subseteq \bar{x}$. Therefore, $\sigma'(\bar{z}_{i2}) \in \mathit{adom}(\mathcal{D})^{|\bar{z}_{i2}|}$. On the other hand, $\mathcal{I} \not\models S_i(\bar{c}'')$, for all $\bar{c}''$. Hence, $\sigma'(\bar{z}_{i2}) \neq \bar{c}''_2$ and $\sigma'(\bar{z}_{i2}) \notin \mathit{adom}(\mathcal{D})^{|\bar{z}_{i2}|}$. Contradiction.

The algorithm then proceeds by looping $\mathbf{O}(\#(\mathcal{D})^{|\bar{y}|})$ times over the $S_i(c'')$s (stored in the $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ registry), checking each time whether, for all $i \in [1, k]$, there exists some "compatible" $\bar{c}''$ s.t. $\mathcal{P}_\mathcal{O} \cup \mathcal{D} \models S_i(\bar{c}'')$ holds. For each atom the algorithm runs a linear **DATALOG** non-deterministic check that uses at most $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ space. Clearly, such a non-deterministic algorithm decides KBQA using, overall, at most $\mathbf{O}(\mathtt{log}\,\#(\mathcal{D}))$ space. □

	(T)CQs		(T)CQs
IS-A_0	**LSpace**	EL-English	**PTime**-complete
IS-A_1	**NLSpace**-complete	IS-A_5	**coNPTime**-complete
IS-A_2	**PTime**-complete	IS-A_6	**coNPTime**-complete
IS-A_3	**PTime**-complete	IS-A_7	**coNPTime**-complete
IS-A_4	**PTime**-complete	DL-English	**coNPTime**-complete

Table 6.3: Summmary of data complexity results.

6.3.1 Minimal Intractable Constructs

We can now individuate the constructs of DL-English, and a fortiori of any controlled language expressing a **coNPTime**-hard ontology language such as $\mathcal{SHIF}$ (as does ACE-OWL) that negatively affect the scalability of controlled language interfaces to OdatabaseASs, namely:

- "only" in subject position (**coNPTime**-hardness of IS-A_5),
- disjunction in predicate position (**coNPTime**-hardness of IS-A_6),
- negation in subject position (**coNPTime**-hardness of IS-A_7).

6.3.2 Maximal Tractable Constructs

The constructs from Figures 6.2 and 6.1 also allow us to identify *maximal* controlled languages contained in DL-English (and a fortiori ACE-OWL) w.r.t. scalability (i.e., tractable data complexity). By merging the (tractable) fragments IS-A_i, for $i \leq 4$, we essentially express the $\mathcal{ELI}$ ontology language (see Chapter 4).

That is, the description logic where negation- and disjunction-free existential concepts are allowed to arbitrarily nest on *both* sides of $\sqsubseteq$. $\mathcal{ELI}$ induces a **PTime**-complete fragment of DL-English, that we term EL-English, which captures most of the constraints and axioms of real-world large-scale biomedical ontologies such as GALEN or SNOWMED [LK07]. We can define EL-English top-down pretty easily by removing from DL-English the grammar rules for negation, disjunction, and universal quantification, and the negative function words. Whence:

Proposition 6.3.3. KBQA *for EL-English and (T)CQs is* **PTime**-*complete in data complexity.*

In such a controlled language arbitrary sentence subordination (and relatives), in combination with, existential quantification and conjunction among **VP**s or **N**s is allowed. Universal quantification is highly controlled and negation and disjunction are ruled out.

6.4 Summary

In this chapter we have proposed several declarative controlled languages, viz., DL-English, EL-English and the IS-A_is, for which KBQA of TCQs is on the one hand *(i)* maximal w.r.t. tractable (**PTime** or less) data complexity and *(ii)* minimal w.r.t. intractable (**NPTime**-hard or more) data complexity. This controlled languages were defined by expressing a space of description logics whose expressiveness lies between that of *DL-Lite* and $\mathcal{ALCI}$.

The strategy adopted was that of expressing first $\mathcal{ALCI}$, a description logic where left and right concepts C, C' occurring to the left and to the right of the subsumption symbol in ontology assertions $C \sqsubseteq C'$ are closed under Boolean operations and symmetrical, and then restricting the syntax of DL-English through subcategorization, to accomodate non-symmetrical and possibly non-Boolean closed left and right concepts, giving way to EL-English and the IS-A_is.

The crux of the intractabiliy results lie in our being able to express (or simulate), on top of *concept disjointness*, i.e., an assertion $C_l \sqsubseteq \neg C_r$ that can be captured sentences of the form "no C_l is a C_r", *concept partitioning*, i.e., an assertion $\neg C_l \sqsubseteq C'_r$, that can be captured by sentences of the form "anybody who is not an C_l is a C'_r".

In general, intractability (w.r.t. data complexity) will arise in every controlled language that induces a partitioning of the data of the OBDAS's database. This will happen whenever their meaning representations can simulate full negation, conjunction and disjunction: the technical **coNPTime**-hardness results from [CdL$^+$06], on which we ground our own work, are based on this intuition. We can thus say that controlled languages like IS-A_5, IS-A_6, IS-A_7 and a fortiori DL-English (which contains them all) are "Boolean closed". EL-English, on the other hand, by being a negation-free fragment of DL-English, remains tractable.

Computational complexity will be even higher if we consider *combined complexity* i.e., when we consider *all* the inputs of KBQA (which however does not provide as accurate a measure for OBDAS efficiency as data complexity). On the other hand, KBSAT for $\mathcal{ALCI}$ and a fortiori KBQA w.r.t. TCQs is **ExpTime**-complete (Recall Theorem 3.3.5 from Chapter 3).

Chapter 7

The Complexity of Pratt and Third's Fragments

In this chapter we study the computational complexity of KBSAT and KBQA for I. Pratt and A. Third's fragments of English [PHT06, PH01, PH04, Thi06], which we outlined in Chapter 4. The fragments of English are interesting in that they can capture many forms of common-sense reasoning such as syllogistic reasoning, that was historically (with Aristotle), the starting point of all research in formal logic. They also overlap in expressiveness with conceptual modelling (e.g., ER-diagrams) and ontology (e.g., OWL, description logics) languages as we saw in Chapter 5.

But just how good would they behave as controlled interface languages? We would like to know to what extent their coverage of English impacts on the data and combined complexity of query evaluation in OBDASs and, in particular, how good they fare as opposed to the description logic-based controlled languages we defined in this thesis, viz., Lite-English, DL-English and the IS-A_is. In particular, data complexity will provide a measure of their scalability. In so doing, we consider

- KBQA w.r.t., on the one hand, TCQs (and their extension to ternary predicate symbols, generalized TCQs) and, on the other hand, arbitrary CQs.
- KBSAT for English and $\mathbf{FO}$ knowledge bases.

Though focused mainly on data complexity, we also take into account combined complexity. In particular, we show which fragments are tractable (**PTime** or less) w.r.t. data complexity for KBQA and KBSAT, intractable (**NPTime** or **coNPTime**-hard) w.r.t. data complexity for KBQA and KBSAT, and undecidable.

Since the fragments of English are, in general, othogonal in expressiveness to ontology languages, only some of the traditional techniques for deriving data complexity bounds are applicable. We therefore adopt the following strategy, that generalizes the *resolution-based saturation decision procedures* from [PHT06] to KBQA and KBSAT: we define a family of resolution procedures that decide several of the fragments of English. Resolution, in general, does not terminate. Joyner in [Jr.76] showed, however, that *(i)* when clauses do not grow beyond a certain depth bound and a certain length bound, i.e., when such bounds exist, it does terminate, and *(ii)* that some sufficient conditions, i.e., several *refinements* of resolution (in its ordered form) induce the existence of such bounds.

When applied to several fragments of $\mathbf{FO}$, whose clauses are subsumed by the $\mathbf{SC}^+$ clause fragment (see [FLHT01], Section 3.5), conditions *(i)* and *(ii)* give rise to resolution decision procedures for those fragments of $\mathbf{FO}$. As we shall see, saturations Γ^∞ for (sets of) clauses Γ from such fragments terminate after polynomially many (deterministic or non-deterministic, depending on the refinements used) steps in the number of individual constants in Γ (i.e., in data complexity).

Pratt and Third show in [PHT06] that we can reduce (un)satisfiability for their fragments to monadic (un)satisfiability. We will see that the resulting monadic clauses are subsumed by $\mathbf{SC}^+$, a class of clauses decidable by resolution. Thereafter, the resolution refinements will allow us to

derive tight data complexity bounds for the **coNPTime** or **NPTime**-hard fragments. As we shall see, undecidability arises when we extend coverage in questions and/or declarations to *arbitrary* pronouns. We also show that for the simpler fragments of English a reduction to DATALOG query evaluation (which is **PTime**-complete w.r.t. data complexity) is possible by defining intermediate fragments of English that we dub *positive fragments*. Other properties of resolution can be exploited to provide reductions to QA (which is in **LSpace** for full FO). The Tables 7.3 and 7.4 summarize the data and combined complexity results obtained. The results from this chapter have been partly published in [TC10b].

We consider as knowledge bases sets $\Gamma \cup \Delta$ of non-ground clauses (i.e., Γ) and ground literals (i.e., Δ): non-ground clauses specify the constraints that hold over the domain (the ontology), whereas ground literals specify the extensional data (the database). We will not distinguish either between the FO meaning representations of a fragment of English and their equisatisfiable clauses resulting from Skolemization and clausification. The integer $\#(\Delta)$ denotes the number of constants in the active domain *adom*(Δ) of Δ (by analogy to description logic knowledge bases).

Formal queries ((U)CQs, (U)TCQs, GCQs) will be accordingly defined over the FO signature $\mathbf{Sig} = (\mathbf{C}, \mathbf{F}, \mathbf{R})$ of such sets $\Gamma \cup \Delta$ of meaning representations, or rather, over their set $\mathbf{R}$ of relations symbols (instead of ontology language role and concept names).

We adopt throughout this chapter ontology language semantic conventions. We assume *(i)* SDA (which implies UNA): frames $\mathbf{Dom}$ consist of (countably infinite) set of individual (pairwise distinct) constant symbols. Thus, domains of interpretation $\mathbb{D}_{\mathcal{I}} \subseteq \mathbf{Dom}$ consist of FO constants (i.e., syntactic entities). Furthermore, given a "database" Δ we assume, for all interpretations $\mathcal{I}$, *(ii)* SNA: $c^{\mathcal{I}} := c$, for each constant $c \in$ *adom*(Δ) that has not been been introduced by Skolemization; observe moreover that *adom*$(\Delta) \subseteq \mathbb{D}_{\mathcal{I}}$.

7.1 First Order Resolution

The resolution calculus was first proposed by Robinson in [Rob65] as a sound and refutation-complete (modulo Skolemization and clausification), backward-chaining deductive calculus for FO. More recently, forward-chaining (a.k.a. saturation-based) variants, have been proposed. Forward chaining variants are useful because they make it easier to define SAT decision procedures for several fragments of FO. Furthemore, unsuccesful saturations can be used to generate minimal (w.r.t. set inclusion and/or the sub-interpretation $\trianglelefteq$ ordering) Herbrand models. The present section is based mainly on [FLHT01] and on [TS00], Chapter 7.

Terms and Clauses. A *term* t is either a variable x, a constant c or a n-ary function symbol $f(t_1, \ldots, t_n)$ applied to n terms. We denote by *Var*(t) and *Ter*(t) the set of its variables and terms, respectively. If *Var*$(t) = \emptyset$, we say that t is *ground*. We denote by *CTer*(t) the set of ground terms of a term t.

A *literal* L is a FO atom. We denote by *Var*(L), *Ter*(L) and *CTer*(L) the set of its variables, terms and closed terms, respectively. Literals are said to be *negative*, written $\overline{L}$ (resp. *positive*, written L) if they are prefixed with an odd (resp. even) number of negations. We write $\pm L$ to denote an arbitrary (i.e., negative or positive) literal. They are said to be *ground* when *Var*$(L) = \emptyset$.

A *clause* C is a disjunction

$$C := L_1 \vee \cdots \vee L_n \vee \overline{L}_{n+1} \vee \cdots \vee \overline{L}_{n+p}$$

of literals. If C consists of one literal, it is said to be a *unit* clause. The clause $\bot$ denotes the *empty clause*, i.e., a clause that has no model. A clause all of whose literals are negative is denoted $\overline{C}$. Given a clause C (resp. a set Γ of clauses), we denote by *Var*(C) (resp. *Var*(Γ)), *Ter*(C) (resp.

Ter(Γ)) and *CTer*(C) (resp. *Ter*(Γ)) its set of variables, terms and closed terms. A clause is said to be *ground* when *Var*$(C) = \emptyset$.

The *depth* $d(t)$ of a term t is recursively defined as follows: *(i)* $d(x) = d(c) := 0$, *(ii)* $d(f(t_1, \ldots, t_n)) := \mathtt{max}\{d(t_i) \mid i \in [1, n]\} + 1$. The *depth* of a literal L is defined as $d(L) := \mathtt{max}\{d(t) \mid t \in \textit{Ter}(L)\}$, the *depth* of a clause C as $d(C) := \mathtt{max}\{d(L) \mid L \text{ is a literal of } C\}$ and the depth of a set Γ of clauses as $d(\Gamma) := \mathtt{max}\{d(C) \mid C \in \Gamma\}$.

Horn Clauses and Definite Programs. If $n \leq 1$, C is said to be a *Horn clause* of *head* L_1 and *body* $\overline{L}_2 \vee \cdots \vee \overline{L}_k$. The fragment of **FO** induced by Horn clauses is denoted **HORN**. Horn clauses are divided into three kinds: *(i) goals*, when $n = 0$ (i.e., the head is empty), *(i) facts*, when $p = 0$ (i.e., the body is empty), and *(ii) rules*.

A set of facts, goals and rules is also known in the literature as a *logic program* and denoted $\mathcal{P}$. By convention, Horn clauses $L_1 \vee \overline{L}_2 \vee \cdots \vee \overline{L}_n$ are written $L_1 \leftarrow L_2 \wedge \cdots \wedge L_n$ and programs $\mathcal{P}$ as

$$\begin{array}{lcll} L_1 & \leftarrow & L_2 \wedge \cdots \wedge L_n & \text{(rules)} \\ & \leftarrow & L_2 \wedge \cdots \wedge L_n & \text{(goals)} \\ L_1 & \leftarrow & & \text{(facts)} \end{array}$$

Horn clauses that contain no function symbols are known as *definite clauses* and their programs as *definite programs*. The fragment they induce is known as (positive) **DATALOG** .

Substitutions and Unifiers. A *unifier* for two terms t and t' is a substitution σ s.t. $t\sigma = t'\sigma$. A unifier σ is said to be a *most general unifier* (mgu) of t and t, when, for every other unifier σ', there exists a renaming σ'' s.t. $\sigma = \sigma'\sigma''$. As a consequence, a most general unifier σ has the property of being unique up to renamings.

The Resolution Calculus. The unrestricted resolution calculus is defined by the rules

$$\textit{res}\ \frac{C \vee \overline{L} \qquad C \vee L'}{(C \vee C')\sigma} \qquad \textit{fact}\ \frac{C \vee L \vee L'}{(C \vee L)\sigma} \qquad (\sigma = \text{mgu}(L, L'))$$

A clause obtained by the application of either *res* or *fact* is called a *resolvent* and the clauses involved the *hypothesis* of the rule.

We give a forward reasoning (a.k.a. saturation-based) account of resolution where we generate *all* the possible clauses C derivable from a set Γ of clauses, i.e., the deductive closure Γ^∞ of Γ under the rules *res* and *fact*. Let ρ denote a function from sets of clauses into sets of clauses that associates to each set Γ its set of (possibly factored) resolvents. Define the *resolution calculus* as a function $\mathcal{R}$ s.t. $\mathcal{R}(\Gamma) := \Gamma \cup \rho(\Gamma)$. Define the *saturation* of Γ by $\mathcal{R}$ inductively by putting

$$\begin{array}{c} \mathcal{R}^0(\Gamma) := \Gamma \\ \mathcal{R}^{k+1}(\Gamma) := \mathcal{R}(\mathcal{R}^k(\Gamma)) \\ \mathcal{R}^\infty(\Gamma) := \bigcup\limits_{n \in \mathbb{N}} \mathcal{R}^i(\Gamma) \end{array}$$

and $\Gamma^\infty := \mathcal{R}^\infty(\Gamma)$.

A *derivation* π of C from Γ is a finite sequence $\Gamma_0, \Gamma_1, \ldots, \Gamma_n$ of sets of clauses, called *goals* or *states* s.t., for all $i \in [1, n]$, $\Gamma_i \subseteq \mathcal{R}(\Gamma_{i-1})$, $\Gamma_0 = \Gamma$ and $C \in \Gamma_n$. If $C = \bot$ we call π a *refutation*. A (saturation) derivation π is said to be *fair* if, for all $i \in \mathbb{N}$ and every pair of clauses $C, C' \in \mathcal{R}^i(\Gamma)$, there exists a $j \geq i$ s.t. $C'' \in \mathcal{R}^j(\Gamma)$ where C'' is the resolvent of C and C'; fairness intuitively says that it is possible to restrict w.l.o.g. attention to non-redundant clauses when searching for a derivation. The positive integer i denotes the size $s(\pi)$ of the derivation.

Furthermore, $\mathcal{R}$ is monotone increasing (w.r.t. $\subseteq$) and Γ^∞ is its fixpoint. The $\cdot^\infty$ operator is monotonic and idempotent (on sets of clauses) i.e., $(\Gamma^\infty)^\infty = \Gamma^\infty$ and if $\Gamma \subseteq \Gamma'$, $\Gamma^\infty \subseteq \Gamma'^\infty$, for all sets Γ, Γ' of clauses.

The $\mathcal{R}$ calculus is known to be sound and refutation complete for $\mathbf{FO}$ (modulo Skolemization and clausification), though not terminating, by observing, on the one hand, that *(i)* entailment can be reduced to unsatisfiability, i.e., $\Gamma \models \varphi$ iff $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable, and, on the other hand, that *(ii)* a $\mathbf{FO}$ formula φ is satisfiable iff φ^{cl}, i.e., the clause resulting from Skolemizing φ and by rewriting it in conjunctive normal form, is satisfiable.

Theorem 7.1.1 (Robinson)**.** *Let Γ be a set of clauses and C a clause. Then:*

1. **(Soundness)** *If $C \in \Gamma^\infty$, then $\Gamma \models C$.*
2. **(Completeness)** *If $\Gamma \models C$, then $\bot \in (\Gamma \cup \{\overline{C}\})^\infty$.*

Given a clause C, the *Herbrand domain* HD_C of C is the set of all the ground terms that can be generated with C's constants and terms. The set

$$HB_C := \{S(t_1, \ldots, t_n) \mid \ R \text{ is a } n\text{-ary relation of } C \text{ and } t_i \in HD_C, \text{ for } i \in [1, n]\}$$

is the *Herbrand base* of C. A *Herbrand interpretation* for C is any interpretation $\mathcal{H} := (\mathbb{D}_\mathcal{H}, \cdot^\mathcal{H})$ with $\mathbb{D}_\mathcal{H} \subseteq HD_C$. By abusing notation a little, it is possible to identify every Herbrand interpretation $\mathcal{H}$ of C with a subset of the Herbrand base of C, i.e., $\mathcal{H} \subseteq HB_C$. This is because the Herbrand models of C are isomorphic to the subsets of HB_C ([Lal97], Proposition IV-5). Hence, for a given C, there are $2^{\#(HB_C)}$ possible Herbrand interpretations. If C contains no functional terms, HD_C and HB_C are finite, infinite otherwise. This notion generalizes also to sets Γ of clauses.

Theorem 7.1.2 (Herbrand)**.** *Let C be a $\mathbf{FO}$ clause. Then the following are equivalent:*

1. *C has a model.*
2. *C has a Herbrand model.*
3. *The set $GR(C)$ of ground clauses of C has a model.*

If $\bot \notin (\Gamma \cup \{\overline{C}\})^\infty$, one can derive from $(\Gamma \cup \{\overline{C}\})^\infty$ a Herbrand model $\mathcal{H}$ of Γ. The Herbrand theorem enforces that the problem of finding a resolution refutation for a clause C from a set Γ reduces, to the problem of showing that the set

$$GR(C) := \{C\sigma \mid \sigma \colon FV(C) \to Con(C)\}$$

of *ground clauses of* C has a model. Clearly, this notion generalizes immediately to sets Γ of clauses.

The $\mathcal{R}$ calculus provides a sound and complete (though not necessarily terminating) query answering algorithm for certain answers.

Corollary 7.1.3. *Let $(\mathcal{O}, \mathcal{D})$ be a knowledge base. Let body $\varphi(\bar{x})$ be a UCQ. Let $\bar{c}$ be a sequence of $|\bar{x}|$ constants. Then the following are equivalent:*

1. *$\bar{c} \in cert(\varphi, \mathcal{O}, \mathcal{D})$.*
2. *$\bot \in (\mathcal{O}^{cl} \cup \mathcal{D}^{cl} \cup \{\overline{\varphi(\bar{c})^{cl}}\})^\infty$.*
3. *$GR(\mathcal{O}^{cl}, \mathcal{D}^{cl}, \overline{\varphi(\bar{c})^{cl}})$ has no model.*

Separation Property. In resolution derivations the order (or strategy) in which clauses are resolved with each other is irrelevant. In particular, we can delay using ground clauses to the last steps of derivations. This property that we call *separation* will be quite useful when analyzing data complexity later on.

Proposition 7.1.4. *Let Γ be a set of non-ground clauses, Δ a set of ground literals and C a clause. Then, if there exists a derivation π of C from $\Gamma \cup \Delta$, we can transform π into a derivation π' where resolutions involving ground literals in Δ are delayed to its last steps.*

Proof. By induction in the size $s(\pi)$ of derivations. If $s(\pi) = 0$, then the property holds trivially, since neither *res* nor *fact* have been applied and either $C \in \Gamma$ or $C \in \Delta$. Suppose by IH that the property holds for every derivation of size k. Let π be a derivation with $s(\pi) = k + 1$. Then π is of the following form:

$$\dfrac{\dfrac{\begin{array}{cc}\Gamma & \Delta \\ \pi_0 & \pi_2 \\ \vdots & \vdots \\ C \vee \overline{L}_1 \vee \overline{L}_0 & L'_0\end{array}}{(C \vee \overline{L}_1)\sigma_0}\,\textit{res} \quad \begin{array}{c}\Gamma \\ \pi_1 \\ \vdots \\ C' \vee L'_1\end{array}}{(C \vee C')\sigma_1}\,\textit{res}$$

with $\sigma_0 = \text{mgu}(L_0, L'_0)$ and $\sigma_1 = \text{mgu}(L_1, L'_1)$. Let π'_0 and π'_1 be the derivations thus obtained. Put $\bar{z} := \mathit{Var}(L_0)$ and $\bar{w} := \mathit{Var}(C, L_1) \setminus \bar{z}$. Let $\bar{c}$ be the constants occuring in the ground literal L'_0. Then, by inspection it turns out that $\sigma_0 = \{\bar{z} \mapsto \bar{c}, \bar{w} \mapsto \bar{w}\}$ and $\sigma_1 = \sigma_0\{\bar{w} \mapsto \bar{t}, \bar{u} \mapsto \bar{t}'\}$, for all variables $\bar{u} \subseteq \mathit{Var}(C\sigma_0, C'\sigma_0)$, where $\bar{t}$ and $\bar{t}'$ are arbitrary (sequences of) terms. By IH the property holds for the subderivations π_0 and π_1 of $C \vee \overline{L}_0 \vee \overline{L}_1$ and $C' \vee L'_1$ from Γ (which are of size $\leq k$). We can thus transform π into the following derivation π' (with subderivations π'_0 and π'_1):

$$\dfrac{\dfrac{\begin{array}{cc}\Gamma & \Gamma \\ \pi'_0 & \pi'_1 \\ \vdots & \vdots \\ C \vee \overline{L}_0 \vee \overline{L}_1 & C' \vee L'_1\end{array}}{(C \vee C' \vee \overline{L}_0)\sigma'_1}\,\textit{res} \quad \begin{array}{c}\Delta \\ \pi_2 \\ \vdots \\ L'_0\end{array}}{(C \vee C')\sigma'_0}\,\textit{res}$$

where $\sigma'_1 := \{\bar{w} \mapsto \bar{t}, \bar{z} \to \bar{z}\}$ and $\sigma'_0 := \sigma'_1\{\bar{z} \mapsto \bar{c}, \bar{u} \to \bar{t}'\}$. Clearly, $\sigma'_0 = \text{mgu}(L_0, L'_0)$ and $\sigma'_1 = \text{mgu}(L_1, L'_1)$. Furthermore, $(C \vee C')\sigma'_0$ is identical to $(C \vee C')\sigma_1$ up to renaming of variables. Therefore, π' is the derivation we were looking for. □

Lemma 7.1.5. *Let Γ be a set of clauses and Δ a set of ground clauses s.t. Γ is satisfiable. If $\bot \in (\Gamma \cup \Delta)^\infty$, then there exists a set $\Gamma' \subseteq \Gamma^\infty$ s.t. (i) $d(\Gamma') \leq d(\Delta)$, (ii) $\bot \in (\Gamma' \cup \Delta)^\infty$ and (iii) Γ' is finite.*

Proof. By Proposition 7.1.4, we know that we can transform any resolution derivation π of a clause C from hypotheses $\Gamma \cup \Delta$ into a derivation π' where resolutions involving ground literals (i.e., ground unit clauses) in Δ have been delayed to the last steps in the derivation. This property holds in particular for refutations. Therefore, if $\bot \in (\Gamma \cup \Delta)^\infty$, there exists an $i \in \mathbb{N}$ s.t. $\bot \in (\mathcal{R}^i(\Gamma) \cup \Delta)^\infty$. Clearly, $\mathcal{R}^i(\Gamma) \subseteq \Gamma^\infty$.

We now claim that there exists a set $\Gamma' \subseteq \mathcal{R}^i(\Gamma)$ s.t. *(i)* Γ' is finite, *(ii)* $\bot \in (\Gamma' \cup \Delta)$ and *(iii)* $d(\Gamma') \leq d(\Delta)$.

Within $\mathcal{R}^i(\Gamma)$ we can distinguish the clauses C that resolve with a unit ground clause $L \in \Delta$ and those that do not. It is from those clauses that $\bot$ will be derived. Moreover, since Δ is a finite set, finitely many such clauses will be used. Unification and a fortiori resolution would fail (by occurs-check or clash) if such clauses C were of depth $> d(\Delta)$. Put $\Gamma' := \{C \in \mathcal{R}^i(\Gamma) \mid C \text{ resolves with some } L \in \Delta\}$. Clearly, Γ' satisfies conditions *(i)–(iii)*. □

Lemma 7.1.6 (Separation)**.** *Let Γ be a set of clauses and Δ a set of ground clauses s.t. Γ is satisfiable. Then, $\bot \in (\Gamma \cup \Delta)^\infty$ iff there exists a set $\Gamma' \subseteq \Gamma^\infty$ s.t. (i) $d(\Gamma') \leq d(\Delta)$, (ii) $\bot \in (\Gamma' \cup \Delta)^\infty$ and (iii) Γ' is finite.*

Proof. The ($\Rightarrow$) direction follows from Lemma 7.1.5. For the ($\Leftarrow$) direction, suppose that such a Γ' exists. Then

$$\begin{array}{lll} (\Gamma' \cup \Delta)^\infty & \subseteq & (\Gamma^\infty \cup \Delta)^\infty \qquad\qquad (\text{by monotonicity}) \\ & \subseteq & ((\Gamma \cup \Delta)^\infty \cup (\Gamma \cup \Delta)^\infty)^\infty \quad (\text{by monotonicity}) \\ & = & ((\Gamma \cup \Delta)^\infty)^\infty \\ & = & (\Gamma \cup \Delta)^\infty \qquad\qquad (\text{by idempotence }) \end{array}$$

holds, whence $\bot \in (\Gamma' \cup \Delta)^\infty$ implies $\bot \in (\Gamma \cup \Delta)^\infty$, which is what we wanted to prove. □

7.2 Refinements of Resolution

As we said, unrestricted resolution, though sound and (refutation) complete, does not provide, per se, decision procedures. A saturation-based derivation may not terminate even when applied to a decidable set Γ of clauses. This is because the *res* and *fact* rules may generate clauses of arbitrary depth and/or length. Decidability can be achieved by considering *refinements* of resolution, in which *(i)* well-founded and substitution-invariant orderings on literals (a.k.a. *acceptable orderings*) constrain the application of the calculus rules and *(ii)* new rules that control clause growth are used. This section derives from results that originate in [Jr.76] and that [BG01, FLHT01] extend.

A-ordered Resolution. A (strict) partial order $\prec$ on literals is an irreflexive and transitive relation on literals. An order $\prec$ on literals is said to be an *acceptable* order (A-order) whenever *(i)* $\prec$ is well-founded (i.e., contains no infinite descending chains $\cdots \prec L_{i-1} \prec L_i \prec \cdots \prec L_0$) and is *liftable* or invariant under substitutions, i.e., if $L \prec L'$, then $L\sigma \prec L'\sigma$, for all σ.

Such orderings can be extended to clauses and sets thereof in the standard way by putting, on the one hand, that $C \prec C'$ whenever for each literal L of C, there exists a literal L' of C' s.t. $L \prec L'$, and, on the other hand $\Gamma \prec \Gamma'$ whenever, for every clause $C \in \Gamma$, there exists a clause $C' \in \Gamma'$ s.t. $C \prec C'$. With $\prec$ one can *restrict* the un-restricted resolution calculus $\mathcal{R}$ by resolving and factoring upon literals and clauses that are maximal w.r.t. $\prec$.

We also define the *relative depth* of variable x w.r.t. terms t and literals L s.t. $x \in \mathit{Var}(L)$ and $x \in \mathit{Var}(t)$ as follows: *(i)* $d(x, x) := 0$, *(ii)* $d(x, f(t_1, \ldots, t_n)) := \mathtt{max}\,\{d(x, t_i) \mid i \in [1, n]\} + 1$ and *(iii)* $d(x, L) := \mathtt{max}\,\{d(x, t) \mid t \in \mathit{Ter}(L)\}$.

Lemma 7.2.1. *Let $\prec$ be an ordering over literals. If $\prec$ is an A-order, then $\mathcal{R}$ restricted by $\prec$ yields a sound and refutation-complete resolution calculus.*

Proof. Robinson showed in [Rob65] that one can prove the refutation-completeness of resolution using *semantic trees* which basically constitute a method for constructing a Herbrand model from a saturation in which $\bot$ does not occur (see [BG01], Section 7.6). A-orderings are known to be compatible with this construction. □

Definition 7.2.2. The orderings $\prec_{a+d}$ and $\prec_d$ over literals (and clauses and sets theoreof) are defined as follows:

$$L \prec_d L' \quad \text{iff} \quad \begin{array}{l} d(L) < d(L'), \\ \mathit{Var}(L) \subseteq \mathit{Var}(L') \text{ and} \\ \text{for all } x \in \mathit{Var}(L), d(x, L) < d(x, L'). \end{array}$$

$$L \prec_{a+d} L' \quad \text{iff} \quad \begin{array}{l} \text{(i) } \mathit{ar}(L) < \mathit{ar}(L') \text{ or} \\ \text{(ii) } \mathit{ar}(L) = \mathit{ar}(L') \text{ and } L \prec_d L'. \end{array}$$

Orderings $\prec_d$ and $\prec_{a+d}$ yield the functions ρ_d, ρ_{a+d}, and the ordered resolution calculi $\mathcal{R}_{a+d}$ and $\mathcal{R}_d$. We say that a resolution calculus $\mathcal{R}$ is contained in a resolution calculus $\mathcal{R}'$, which we express in symbols by $\mathcal{R} \subseteq \mathcal{R}'$ (thus overloading the set containment symbol) whenever every derivation π in $\mathcal{R}$ of a clause C from Γ can be transformed into a derivation π' in $\mathcal{R}'$ of C from Γ.

Proposition 7.2.3. *The inclusions $\prec_{a+d} \subseteq \prec_d$ and $\mathcal{R}_{a+d} \subseteq \mathcal{R}_d \subseteq \mathcal{R}$ hold.*

Proof. Immediate from the definition. □

Theorem 7.2.4. *The orderings $\prec_{a+d}$ and $\prec_d$ are well-founded and liftable.*

Proof. Immediate from the definition. □

By applying Lemma 7.2.1 we immediately derive:

Corollary 7.2.5. *The ordered resolution calculi $\mathcal{R}_{a+d}$ and $\mathcal{R}_d$ are sound and refutation-complete.*

A-orderings are important in that, on the one hand, they reduce the search space for resolution derivations (when resolution is a decision procedure) and, on the other hand, they limit the depth of clauses.

A literal L is said to be *covering* if L contains only variables or constants, or, if it contains a function term t, then $\mathit{Var}(t) = \mathit{Var}(L)$. A clause C is said to be *covering* when, for all $L \in \mathit{Lit}(C)$, L is covering. Given two covering clauses C and C', the depth $d(C'')$ of their $\prec_d$-ordered resolvent C'' will be bounded by the depth or their hypothesis, viz., $d(C'') \leq \mathtt{max}\{d(C), d(C')\}$, i.e., as long as covering is preserved, depth does not grow (see [BG01], Lemma 3.6).

Resolution with Splitting and Condensation. Ordered resolution alone, while preventing (by resolving upon covering literals) arbitrary nesting of function terms and thus arbitrarily deep terms, literals and clauses, may not prevent clause length growth.

We need to introduce two more refinements that prevent deriving arbitrarily long clauses, viz., the *splitting* rule *split*, and the *condensation* rule *cond*. The latter deletes repeated literals in clauses belonging to a saturation state. The former splits saturations into subsaturations where clauses have been split, provided that their literals share no variables. These rules are sound and refutation-complete:

$$\mathit{split}\ \frac{\begin{array}{ccc} C \vee L & & C \vee L' \\ \vdots & & \vdots \\ C \vee L \vee L' & C'\sigma & C'\sigma \end{array}}{C'\sigma}\ (\bar{x} \cap \bar{x}' = \emptyset,\ \sigma \text{ mgu}) \qquad \mathit{cond}\ \frac{C \vee L \vee L}{C \vee L}$$

where $\bar{x} = \mathit{Var}(L)$ and $\bar{x}' = \mathit{Var}(L')$ (i.e., L and L' must not share any variables). Notice that these two rules are *order independant* (A-orders affect only *res* and *fact*).

Proposition 7.2.6. *Resolution and A-ordered resolution with splitting and condensation are sound and refutation-complete.*

Proof. Joyner shows in [Jr.76], Theorem 7.2 that adding the condensation rule to a *complete* (w.r.t. semantic trees) resolution calculus, results in a complete resolution calculus. Similarly, it is known (see [FLHT01], Section 2) that splitting preserves completeness. These facts, combined with Corollary 7.2.5 imply this result. □

Monadizations. Monadizations, introduced by Joyner in [Jr.76] are yet another refinement of resolution that ensures a finite bound to the depth (some particular classes of) clauses when applied in combination with resolution saturations.

Let t be a term. A literal L is said to be

- *essentially monadic on* t whenever *(i)* $t \in \mathit{Ter}(L)$ and *(ii)* for all $t' \in \mathit{Ter}(L)$, either $t' = t$ or $t = c$, for some constant c
- *almost monadic on* t whenever *(i)* $t \in \mathit{Ter}(L)$, *(ii)* t is of the form $t = f(t_1, \ldots, t_n)$ and *(iii)* for all $t' \in \mathit{Ter}(L)$, either $t' = t$, $t = c$, for some constant c or $t = x$, for some $x \notin \mathit{Var}(t)$.

Monadizations are defined on almost and essentially monadic literals and clauses. Two previous notions are needed. Let $\mathbf{C}$ be a finite set of constants. On the one hand, we define the set of the *relativized substitutions* of a $\mathbf{Fo}$ formula φ to *Con* of constants and a term t

$$\Sigma^{\mathbf{C}}_{t,\varphi} := \{\sigma \mid \sigma \colon \mathit{Var}(\varphi) \setminus \mathit{Var}(t) \to \{t\} \cup \mathbf{C}\}$$

and, on the other hand, the set of *relativized variants* of φ to $\mathbf{C}$ and t

$$\varepsilon(t, \varphi, \mathbf{C}) := \{\varphi\sigma \mid \sigma \in \Sigma^{\mathbf{C}}_{t,\varphi}\}.$$

We can now define, for every clause C, the *monadization* of clause C w.r.t. $\mathbf{C}$ as follows:

- if for every $L \in \mathit{Lit}(C)$, L is almost monadic on $t \in \mathit{Ter}(L)$, we set

$$\mu(C, \mathbf{C}) := \varepsilon(t, C, \mathbf{C}),$$

- if C is function free and there is exactly one $x \in \mathit{Var}(C)$ s.t. for every $L \in \mathit{Lit}(C)$ with $\#(\mathit{Var}(L)) \geq 2$, $x \in \mathit{Var}(L)$, we set

$$\mu(C, \mathbf{C}) := \varepsilon(x, C, \mathbf{C}),$$

- otherwise we set

$$\mu(C, \mathbf{C}) := \{C\}.$$

Finally, for every set Γ of clauses we define the *monadization* of Γ as

$$\mu(\Gamma) := \bigcup_{C \in \Gamma} \mu(C, \mathit{Con}(\Gamma)). \qquad \textbf{(Mon)}$$

Proposition 7.2.7 ([Jr.76], Theorem 9.1)**.** *Resolution and A-ordered resolution with monadization are sound and refutation-complete.*

The key idea of monadizations is to transform sets of non-covering clauses which are, however, almost and/or essentially monadic clauses can be transformed ("monadized") into sets of covering clauses. This, we know, entails the existence of a depth $d \in \mathbb{N}$ bound. This expedient, combined with A-ordered resolution (the $\prec_d$ ordering) condensantion and possibly splitting gives rise to several resolution saturation-based decision procedures.

		split	*mon*	*cond*	*cond mon*	*split cond*	*split mon*	*split mon cond*
	$\mathcal{R}_{1,1}$	$\mathcal{R}_{1,2}$	$\mathcal{R}_{1,3}$	$\mathcal{R}_{1,4}$	$\mathcal{R}_{1,5}$	$\mathcal{R}_{1,6}$	$\mathcal{R}_{1,7}$	$\mathcal{R}_{1,8}$
d	$\mathcal{R}_{2,1}$	$\mathcal{R}_{2,2}$	$\mathcal{R}_{2,3}$	$\mathcal{R}_{2,4}$	$\mathcal{R}_{2,5}$	$\mathcal{R}_{2,6}$	$\mathcal{R}_{2,7}$	$\mathcal{R}_{2,8}$
$d+a$	$\mathcal{R}_{3,1}$	$\mathcal{R}_{3,2}$	$\mathcal{R}_{3,3}$	$\mathcal{R}_{3,4}$	$\mathcal{R}_{3,5}$	$\mathcal{R}_{3,6}$	$\mathcal{R}_{3,7}$	$\mathcal{R}_{3,8}$

Table 7.1: Resolution calculi.

Resolution Decision Procedures. By combining *res* and *fact* with the A-orders $\prec_d$ and $\prec_{d+a}$, the *split* and *cond* rules and the monadization refinement we obtain a family of 24 possible calculi, that we spell out in Table 7.1. We define the saturation of Γ^∞ by any of the resolution calculi, for $i \in [1,3], j \in [1,8]$ inductively as follows:

$$\begin{aligned}\mathcal{R}^0_{i,j}(\Gamma) &:= \Gamma,\\ \mathcal{R}^{k+1}_{i,j}(\Gamma) &:= \mathcal{R}_{i,j}(\mathcal{R}^k_{i,j}(\Gamma)),\\ \mathcal{R}^\infty_{i,j}(\Gamma) &:= \bigcup_{k\in\mathbb{N}} \mathcal{R}^k_{i,j}(\Gamma),\end{aligned}$$

whence $\Gamma^\infty := R^\infty_{i,j}(\Gamma)$. Denote by κ the function that maps sets of clauses into their consensed sets. Let $\Gamma := \Gamma' \cup \{C \vee L \vee L'\}$, $\Gamma_1 := \Gamma' \cup \{C \vee L\}$ and $\Gamma_2 := \Gamma' \cup \{C \vee L'\}$ be sets of clauses where $\mathit{Var}(L) \cap \mathit{Var}(L') = \emptyset$. Then $\mathcal{R}_{2,8}$ will be, for instance, defined as

$$\mathcal{R}_{2,8}(\Gamma) := (\kappa(\mu(\mathcal{R}_d(\Gamma_1))) \cup \Gamma) \cup (\kappa(\mu(\mathcal{R}_d(\Gamma_2))) \cup \Gamma). \qquad (\mathcal{R}_{2,8})$$

Notice that whenever we make use of the splitting rule, saturation derivations π become trees. Consider $\mathcal{R}_{2,8}$. A derivation in this calculus is a tree of root $\Gamma = \Gamma_{0,1}$, where every internal node $\Gamma_{i,j}$, for $i, j \in \mathbb{N}$, has at most two siblings, $\Gamma_{i+1,2^j-1}$ and $\Gamma_{i+1,2^j}$, s.t. $\Gamma_{i+1,2^j-1} \cup \Gamma_{i+1,2^j} \subseteq \mathcal{R}^{i+1}_{2,8}(\Gamma)$. Each such node is called a *state*. See Figure 7.1.

A derivation π is said to be a refutation whenever, for every leaf state $\Gamma_{i,j}$, $\bot \in \Gamma_{i,j}$. The integer $i \in \mathbb{N}$ is called the *depth* or *rank* of the (saturation) tree.

Notice that without splitting saturations are linear (i.e., a sequence). Notice also that a non-refutation provides a way for specifying a model of Γ. Finally, observe that $\mathcal{R} = \mathcal{R}_{1,1}$, $\mathcal{R}_d = \mathcal{R}_{2,1}$ and $\mathcal{R}_{d+a} = \mathcal{R}_{3,1}$.

Decidable classes. In this section we recall the definition of the class $\mathbf{SC}^+$ of clauses for which A-ordered resolution with *all* the refinements, viz., monadization, splitting and condensation, constitutes a decision procedure.

The $\mathbf{SC}^+$ class extends the Ackermann $\exists^*\forall\exists^*$ and Gödel $\exists^*\forall^2\exists^*$ $\mathbf{FO}$ fragments (see [FLHT01], Section 3.5). In particular, $\exists^*\forall\exists^*$ can be decided without using the splitting rule [Jr.76]. We recall that a $\exists^*\forall\exists^*$- or $\exists^*\forall^2\exists^*$- formula is a formula $\varphi := \exists x_1 \cdots \exists x_n \forall x_{n+1} \exists x_{n+2} \cdots \exists x_{n+m} \psi$, resp. $\varphi := \exists x_1 \cdots \exists x_n \forall x_{n+1} \forall x_{n+2} \exists x_{n+3} \cdots \exists x_{n+m} \psi$, in prenex normal form with $n, m \geq 0$ and ψ quantifier-free.

Definition 7.2.8. The class $\mathbf{SC}^+$ of clauses is the class where every C satisfies

- $\mathit{Var}(C) = \mathit{Var}(t)$, for every functional term $t \in \mathit{Ter}(C)$, and
- either $\#(\mathit{Var}(L)) \leq 1$ or $\mathit{Var}(L) = \mathit{Var}(C)$, for all $L \in \mathit{Lit}(C)$.

Theorem 7.2.9 ([FLHT01], Theorem 3.25 and [Jr.76], Corollary 9.3)**.** *The resolution $\mathcal{R}_{2,8}$ calculus is a decision procedure for $\mathbf{SC}^+$ and the $\exists^*\forall^2\exists^*$ fragment. Similarly, the resolution $\mathcal{R}_{2,6}$ calculus is a decision procedure for the $\exists^*\forall\exists^*$ fragment.*

7.3 Resolution Decision Procedures and Data Complexity

Given a finite signature $\mathbf{Sig} = (\mathbf{C}, \mathbf{F}, \mathbf{R})$ with $\#(\mathbf{C})$ constants and a set Γ of clauses over this signature beloging to a class decidable by resolution, we show that we can check for unsatisfiability of Γ in time deterministic polynomial in $\#(\mathbf{C})$ (if we omit splitting) or in time non-deterministic polynomial in $\#(\mathbf{C})$.

Lemma 7.3.1. *Let* $\mathbf{Sig} = (\mathbf{C}, \mathbf{F}, \mathbf{R})$ *be a first order finite signature. Consider a clause set* Γ *over a (finite) set* $\mathbf{V}$ *of variables and suppose there exist both a term depth bound* $d \in \mathbb{N}$ *and a clause length bound* $k \in \mathbb{N}$*. Then*

1. *the number of clauses derivable by the saturation is*
   - i. *exponential in* $\#(\mathbf{C})$ *and double exponential in* $\max\{\#(\mathbf{F}), \#(\mathbf{R})\}$*, if we use the splitting rule, or*
   - ii. *polynomial in* $\#(\mathbf{C})$ *and exponential in* $\max\{\#(\mathbf{F}), \#(\mathbf{R})\}$ *otherwise,*
3. *the depth of the saturation is polynomial in* $\#(\mathbf{C})$ *and exponential in* $\max\{\#(\mathbf{F}), \#(\mathbf{R})\}$*, and*
4. *there are polynomially many in* $\#(\mathbf{C})$ *and exponentially many in* $\max\{\#(\mathbf{F}), \#(\mathbf{R})\}$ *clauses of length* $\leq k$*.*

Proof. Assume that a nesting depth bound d exists for terms and a length bound k for clauses and which are independent of c. Consider now the following parameters:

- c is the number $\#(\mathbf{C})$ of constant symbols in $\mathbf{C}$,
- v is the number $\#(\mathbf{V})$ of variables in $\mathbf{V}$,
- f is the number $\#(\mathbf{F})$ of function symbols in $\mathbf{F}$,
- p is the number $\#(\mathbf{R})$ of predicate symbols in $\mathbf{R}$,
- the maximum arity of the function symbols is ar_f, and
- the maximum arity of the predicate symbols is ar_p.

We can define the maximal number t_i of terms of depth i inductively by setting

$$\begin{aligned} t_0 &:= v + c \\ t_{i+1} &:= f \cdot t_n^{ar_f} \end{aligned}$$

and, since $i \leq d$, derive an upper bound to the number t of terms of depth $\leq d$

$$\begin{aligned} t \ &\leq \textstyle\sum_{i=0}^{d} t_i \\ &= t_0 + (f \cdot t_0^{ar_f}) + \cdots + (f \cdot t_{d-1}^{ar_f}) \\ &= f^0 \cdot (v+c)^{ar_f^0} + \cdots + f^d \cdot (v+c)^{ar_f^d} \\ &:= \mathtt{p_t}(c) \end{aligned} \tag{7.1}$$

thus defining a polynomial $\mathtt{p_t}(c)$ of degree $\deg(\mathtt{p_t}) \leq ar_f^d$. This in its turn yields as upper bound to the number l of positive and negative literals

$$\begin{aligned} l \ &\leq 2 \cdot p \cdot t^{ar_p} \\ &\leq 2 \cdot p \cdot \mathtt{p_t}(c)^{ar_p} \\ &:= \mathtt{p_l}(c) \end{aligned} \tag{7.2}$$

thus defining a polynomial $\mathtt{p_l}(c)$ of degree $\deg(\mathtt{p_l}) \leq ar_f^d \cdot ar_p$. Finally, from l we derive an upper bound to the number *cl* of clauses of length $\leq k$

$$\begin{aligned} cl \ &\leq l^k \\ &\leq \mathtt{p_l}(c)^k \\ &:= \mathtt{p_{cl}}(c) \end{aligned} \tag{7.3}$$

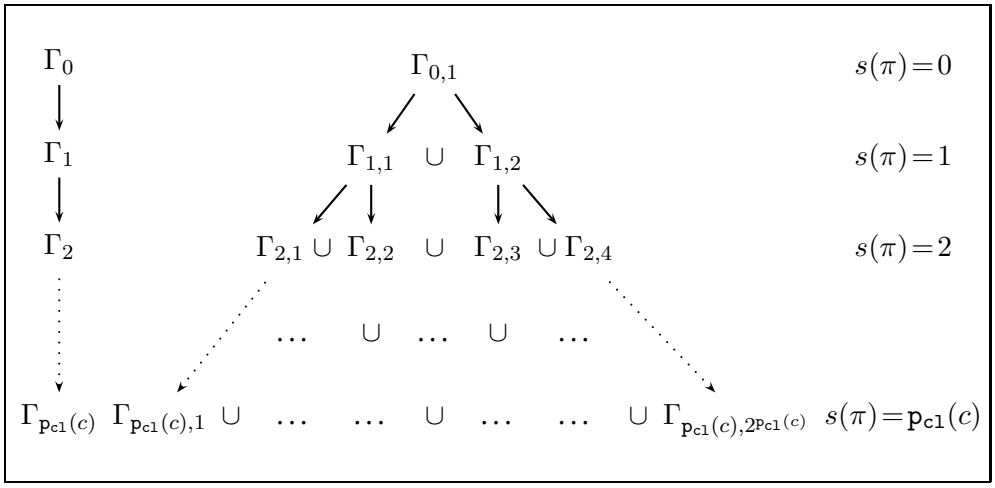


Figure 7.1: A saturation π with and without splitting.

which again defines a polynomial $\mathtt{p_{cl}}(c)$ of degree $\deg(\mathtt{p_{cl}}) \leq ar_f^d \cdot ar_p \cdot k$.

The splitting rule splits the states of a saturation derivation into two separate states, thus yielding a derivation tree of branching factor 2, depth $\leq \mathtt{p_{cl}}(c)$, states of size $\leq \mathtt{p_{cl}}(c)$ and overall (worst-case) size $\leq 2^{\mathtt{p_{cl}}(c)}$. If we omit the splitting rule, the saturation, now linear, stops after deriving a a sequence of $\leq \mathtt{p_{cl}}(c)$ states, each of size $\leq \mathtt{p_{cl}}(c)$.

Set now $s := \mathtt{max}\{v, f, p, ar_f, ar_p, d, k\}$. Clearly, as the reader can see: *(i)* we can derive from (7.1) an exponential function $\mathtt{exp_t}$ of base s and exponent s s.t. $t \leq \mathtt{exp_t}(s, s)$; *(ii)* we can derive from (7.2) an exponential function $\mathtt{exp_1}$ of base s and exponent s s.t. $l \leq \mathtt{exp_t}(s, s)$; and *(iii)* we can derive from (7.3) an exponential function $\mathtt{exp_{cl}}$ of base s and exponent s s.t. $cl \leq \mathtt{exp_{cl}}(s, s)$.

This in its turn entails that π is of depth $\leq \mathtt{exp_{cl}}(s, s)$, has states of maximum size $\leq \mathtt{exp_{cl}}(s, s)$ and stops after generating either $\leq 2^{\mathtt{exp_{cl}}(s,s)}$ or $\leq \mathtt{exp_{cl}}(s, s)$ states in case π is built using or not using the splitting rule, depending in the $\mathcal{R}_{i,j}$s, for $i \in [1, 4], j \in [1, 8]$. See Figure 7.1. Notice that the bound $2^{\mathtt{exp_{cl}}(s,s)}$ is a double exponential in s and may not be optimal. □

Theorem 7.3.2. KBSAT *is in* **NPTime** *in data complexity for* $\mathbf{SC}^+$, $\exists^*\forall\exists^*$ *and* $\exists^*\forall^2\exists^*$.

Proof. Let $\Sigma \cup \Delta$ be a set of $\mathbf{SC}^+$ clauses (resp. $\exists^*\forall\exists^*$ or $\exists^*\forall^2\exists^*$ formulas). Consider now a $\mathcal{R}_{2,5}$-saturation. Calculus $\mathcal{R}_{2,5}$ decides $\mathbf{SC}^+$ and saturations finitely converge. Assume w.l.o.g. that Σ contains no constants and that Δ is of depth $d(\Delta) = 0$ and has $\#(\Delta)$ distinct constants. By Lemma 7.3.1 we know that the saturation will be tree-shaped, of rank $\leq \mathtt{p}(\#(\Delta))$, of size $\leq 2^{\mathtt{p}(\#(\Delta))}$ and of maximal state of size $\leq \mathtt{p}(\#(\Delta))$.

Outline a non-deterministic algorithm for KBSAT as follows. Start with $\Sigma \cup \Delta$. For each rank $i \in [0, \mathtt{p}(\#(\Delta))]$ of the saturation, guess/choose a state $j \in [0, 2^i]$. Notice that the algorithm will make polynomially many choices in $\#(\Delta)$. Finally, check, in time polynomial in $\#(\Delta)$ whether $\bot$ is in the resulting state, and, if no, compute, in time polynomial in $\#(\Delta)$, a Herbrand model of $\Sigma \cup \Delta$. □

7.4 The Positive Fragments

Before looking in detail at KBQA and KBSAT for the fragments of English, we study a simpler case, viz., the case of the *positive fragments*. These fragments are defined by eliminating negation,

FO fragment	Data complexity of KBSAT
$\mathbf{S}^+$	in **NPTime** [Th 7.3.2]
$\exists^*\forall\exists^*$	in **NPTime** [Th 7.3.2]
$\exists^*\forall^2\exists^*$	in **NPTime** [Th 7.3.2]

Table 7.2: KBSAT data complexity upper bounds for $\mathbf{S}\,\mathbf{C}^+$, $\exists^*\forall\exists^*$ and $\exists^*\forall^2\exists^*$.

i.e., the generalized negative determiner "no" of meaning representation, we recall,

$$\lambda P.\lambda Q.\forall x(P(x) \Rightarrow \neg Q(x)),$$

expressing *set disjointness* and the Boolean operator "is not" or "does not" for negating **VP**s, of meaning representation

$$\lambda P.\neg P$$

from COP, COP+TV, COP+DTV and COP+TV+DTV. This gives rise to the following classes of meaning representations:

– IS-A$^{\forall}_{\exists}$ meaning representations are defined as follows:

$$P(c)$$
$$Q_1x_1(P(x_1), Q(x_1))$$

– IS-A$^{\forall}_{\exists}$+TV meaning representations contain, in addition:

$$S(c, c')$$
$$Q_1x_1(P(x_1), \chi(x_1))$$
$$Q_1x_1(P(x_1), Q_2x_2(Q(x_2), \chi(x_1, x_2)))$$

– IS-A$^{\forall}_{\exists}$+DTV meaning representations contain, in addition:

$$T(c, c', c'')$$
$$Q_1x_1(P(x_1), \zeta(x_1))$$
$$Q_1x_1(P(x_1), Q_2x_2(Q(x_2), \zeta(x_1, x_2)))$$
$$Q_1x_1(P(x_1), Q_2x_2(Q(x_2), Q_3x_3(N(x_3), \zeta(x_1, x_2, x_3))))$$

where *(i)* P, Q, N, etc., denote unary predicate symbols, *(ii)* S, S', etc., binary symbols, *(iii)* T, T', etc., ternary symbols, *(iv)* $Q_ix_i(\varphi, \psi)$ stands for either $\forall x_i(\varphi \Rightarrow \psi)$ or $\exists x_i(\varphi \wedge \psi)$, *(v)* $\chi(\bar{x})$ denotes a binary atom where $\bar{x}$ occurs free, and *(vi)* $\zeta(\bar{x})$ denotes a ternary atom where $\bar{x}$ occurs free. Finally, IS-A$^{\forall}_{\exists}$+TV+DTV is the fragment that subsumes them all. The reader will find a diagram describing their relative expressiveness in Figure 7.3.

Theorem 7.4.1. KBQA *is in* **LSpace** *in data complexity and* **NPTime***-complete in combined complexity for IS-A$^{\forall}_{\exists}$ and (U)CQs.* KBSAT *is also in* **LSpace** *in data complexity.*

Proof. By definition, IS-A$^{\forall}_{\exists}$ is contained in COP. By Theorem 4.1.15 we know that COP is contained in *DL-Lite* and a fortiori by Lite English. The result follows from this combined with Corrollaries 4.3.2 and 4.3.6. Membership in **LSpace** for KBSAT follows from results for *DL-Lite* knowledge bases [CdV$^+$06]. □

Lemma 7.4.2. *Let $\Gamma \cup \Delta$ be a set of IS-A$^{\forall}_{\exists}$+TV meaning representations and φ a UCQ. We can transform $\Gamma \cup \Delta$ in time polynomial on $\#(\Gamma)$ into a set $\Gamma' \cup \Delta'$ of* **FO** *$\forall^*\exists^*$-sentences s.t. cert$(\varphi, \Gamma, \Delta) =$ cert$(\varphi, \Gamma', \Delta')$.*

Proof. Transform $\Gamma \cup \Delta$ into a set $\Gamma' \cup \Delta'$ of $\forall^*\exists^*$-sentences of the form $\forall\bar{x}(\varphi(\bar{x}) \Rightarrow \psi(\bar{x}))$ and $\forall\bar{x}(\varphi(\bar{x}) \Rightarrow \exists\bar{y}\psi(\bar{x},\bar{y}))$. Start by putting $\Gamma' := \emptyset$ and $\Delta' := \Delta$. Next, for each $\chi \in \Gamma$,

- if $\chi = \forall x(P(x) \Rightarrow Q(x))$, then $\Gamma' := \Gamma' \cup \{\chi\}$,
- if $\chi = \exists x(P(x) \wedge Q(x))$, then $\Delta' := \Delta' \cup \{P(c), Q(c)\}$,
- if $\chi = \forall x(P(x) \Rightarrow \varphi(x))$, then $\Gamma' := \Gamma' \cup \{\chi\}$,
- if $\chi = \exists x(P(x) \wedge \varphi(x))$, then $\Delta' := \Delta' \cup \{P(c'), \varphi(c')\}$,
- if $\chi = \forall x(P(x) \Rightarrow \exists y(S(x,y) \wedge Q(y)))$, then $\Gamma' := \Gamma' \cup \{\chi\}$,
- if $\chi = \exists x(P(x) \wedge \forall y(S(x,y) \Rightarrow Q(x)))$, then
  - $\Delta' := \Delta' \cup \{P(d)\}$ and
  - $\Gamma' := \Gamma' \cup \{\forall y(S(c',y) \Rightarrow Q(x))\}$.

Where c' is a fresh Skolem constant. This rewriting procedure for $\Gamma \cup \Delta$ introduces $\leq \#(\Gamma)$ new Skolem constants $c'_1, \ldots, c'_{\#(\Gamma)}$ and proceeds in time linear in $\#(\Gamma)$. We now claim that

$$\Gamma' \cup \Delta' \models \varphi(\bar{c}) \quad \text{iff} \quad \Gamma \cup \Delta \models \varphi(\bar{c}). \tag{$\dagger$}$$

($\Rightarrow$) Assume $\Gamma' \cup \Delta' \models \varphi(\bar{c})$ and let $\mathcal{I} \models \Gamma \cup \Delta$. Then we can define an arbitrary expansion $\mathcal{I}'$ of $\mathcal{I}$ s.t. $\mathcal{I}' \models \Gamma' \cup \Delta'$, which implies, by hypothesis, that $\mathcal{I}' \models \Gamma \cup \Delta$ too. Now, if $\mathcal{I}'$ is a Skolem expansion of $\mathcal{I}$, then $\mathcal{I}$ is then the restriction of $\mathcal{I}'$ to the language/signature of $\mathcal{I}$. Therefore, $\mathcal{I}$ and $\mathcal{I}'$ coincide on $\varphi(\bar{c})$, whence $\mathcal{I} \models \varphi(\bar{c})$.

($\Leftarrow$) Assume $\Gamma \cup \Delta \models \varphi(\bar{c})$ and let $\mathcal{I}'$ be an arbitrary interpretation s.t. $\mathcal{I}' \models \Gamma' \cup \Delta'$. Since $\Gamma' \cup \Delta'$ is a (partial) Skolem theory, the models of $\Gamma' \cup \Delta'$ are contained by the models of $\Gamma \cup \Delta$, whence $\mathcal{I}' \models \Gamma \cup \Delta$ and, therefore, $\mathcal{I}' \models \varphi(\bar{c})$. □

In what follows we perform a reduction to KBQA in the **DATALOG** query language, which is known to be in **PTime** in data complexity (see [EGDV01], Theorem 4.4). To this purpose, we will define a structure that, although not being necessarily a model of a knowledge base, can be, however, homomorphically embedded into its models. Since we are interested in answering UCQs, this suffices, because, as we saw in Theorem 3.4.2 from Chapter 3, UCQs are closed under homomorphisms. This structure is defined using the $\sim_\Gamma$ equivalance relation on $\mathbf{Dom}$ that we introduce below.

Definition 7.4.3. Let Γ be a finite set of existentially quantified **FO** formulas. Define an equivalence relation $\sim_\Gamma$ on $\mathbf{Dom}$ by putting:

$c \sim_\Gamma c'$ iff there exists an interpretation $\mathcal{I}$, a formula $\exists y\varphi \in \Gamma$ and assignments γ, γ' s.t.
(i) $\gamma(y) = c$, (ii) $\gamma'(y) = c'$, (iii) $\mathcal{I}, \gamma \models \varphi$ and (iv) $\mathcal{I}, \gamma' \models \varphi$.

We denote by $[c]$ the equivalence class of c by $\sim_\Gamma$, i.e., the set $\{c' \in \mathbf{Dom} \mid c \sim_\Gamma c'\}$.

This yields (at most) the following $\#(\Gamma)$ equivalence classes: $[c_1]$,…,$[c_{\#(\Gamma)}]$. We denote by $\mathbf{Dom}/_{\sim_\Gamma}$ the quotient set of $\mathbf{Dom}$ by $\sim_\Gamma$. Set $\mathbf{Dom}' := \mathbf{Dom} \cup \mathbf{Dom}/_{\sim_\Gamma}$. In Lemma 7.4.4 we consider assignments γ that map variables to elements in $\mathbf{Dom}'$ and interpretations $\mathcal{I} = (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ where $\mathbb{D}_\mathcal{I} \subseteq \mathbf{Dom}'$. We will also apply the (countable version of the) so-called *axiom of choice* (**AC**) of set theory (see [CL03], Vol. 2, Chapter 7), stated thus:

> Let $\{X_i\}_{i\geq 0}$ be a family of sets. If $i > 0$ and $X_i \neq \emptyset$, for all $i \geq 1$, then there exists a function $\mathbf{c}\colon \mathbb{N} \to \bigcup\{X_i \mid i \geq 1\}$, known as a *choice function*, s.t. $\mathbf{c}(i) \in X_i$, for all $i \geq 1$. **(AC)**

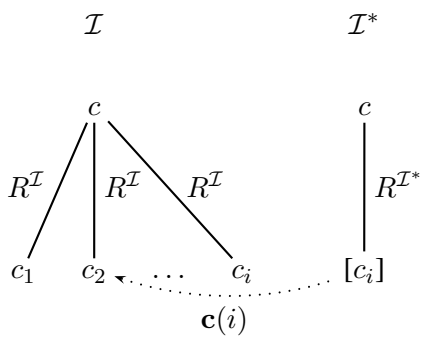


Figure 7.2: The interpretation $\mathcal{I}^*$.

Lemma 7.4.4. *Let $\Gamma \cup \Delta$ be a set of IS-A$^{\forall}_{\exists}$+TV meaning representations and $\varphi(\bar{x})$ a UCQ. We can construct an interpretation $\mathcal{I}^*$ over $\mathbf{Dom}'$ s.t., for all sequence $\bar{c}$ of $|\bar{x}|$ constants, $\mathcal{I}^* \models \varphi(\bar{c})$ iff $\bar{c} \in$ cert$(\varphi, \Gamma, \Delta)$.*

Proof. Let $\Gamma' \cup \Delta'$ be as in Lemma 7.4.2. Consider the sentences in Γ. There are $\leq \#(\Gamma')$ sentences $\varphi_1, \ldots, \varphi_{\#(\Gamma')}$ of the form $\varphi_i := \forall x(\varphi(x) \Rightarrow \exists y \psi(x, y))$. Define a finite interpretation $\mathcal{I}^*$ by putting

$$\begin{aligned} R^{\mathcal{I}^*} &:= \{(c, [c']) \in \mathit{adom}(\Delta') \times \mathbf{Dom}/_{\sim_\Gamma} \mid \Gamma' \cup \Delta' \models R(c, c'), i \in [1, p]\} \\ &\quad \cup \{(c, c') \in \mathit{adom}(\Delta') \times \mathit{adom}(\Delta') \mid \Gamma' \cup \Delta' \models R(c, c')\}, \\ P^{\mathcal{I}^*} &:= \{[c'] \in \mathbf{Dom}/_{\sim_\Gamma} \mid \Gamma' \cup \Delta' \models P(c')\} \\ &\quad \cup \{c' \in \mathit{adom}(\Delta') \mid \Gamma' \cup \Delta' \models P(c')\}, \end{aligned}$$

for every unary predicate P and every binary predicate R in the signature of $\Gamma \cup \Delta$. It is of domain $\mathbb{D}_{\mathcal{I}^*} := \mathit{adom}(\Delta') \cup \mathbf{Dom}/_{\sim_\Gamma}$. We claim that this finite interpretation $\mathcal{I}^*$, that is not necessarily a model of $\Gamma' \cup \Delta'$, is such that

$$\Gamma' \cup \Delta' \models \varphi(\bar{c}) \quad \text{iff} \quad \mathcal{I}^* \models \varphi(\bar{c}). \tag{$\dagger$}$$

($\Rightarrow$) We claim that $\mathcal{I}^*$ is a model of $\Gamma' \cup \Delta'$. Clearly, by definition, $\mathcal{I}^* \models \Delta'$. Consider now $\chi \in \Gamma'$. If $\chi \neq \varphi_i$, $\mathcal{I}^* \models \chi$. Otherwise, when $\chi = \varphi_i := \forall x(\varphi(x) \Rightarrow \exists y \psi(x, y))$, then, let $\gamma \colon \{x\} \to \mathbb{D}_{\mathcal{I}^*}$ be an assignment s.t. $\mathcal{I}^*, \gamma \models \varphi(x)$. Observe that $\psi(x, y) := A(y) \wedge R(x, y)$. Let c be the constant s.t. $\gamma(x) = c$. Define now an assignment $\gamma' \colon \{x, y\} \to \mathbb{D}_{\mathcal{I}^*}$ by putting *(i)* $\gamma'(x) := \gamma(x)$ and *(ii)* $\gamma(y) := [c']$ iff $\Gamma' \cup \Delta' \models A(c') \wedge R(c, c')$. Clearly, $\gamma'(y) \in A^{\mathcal{I}^*}$ and $(\gamma'(x), \gamma'(y)) \in R^{\mathcal{I}^*}$, i.e., $\mathcal{I}^*, \gamma' \models \psi(x, y)$ and a fortiori $\mathcal{I}^* \models \chi$ and $\mathcal{I}^* \models \Gamma'$. Therefore, $\mathcal{I}^* \models \varphi(\bar{c})$.

($\Leftarrow$) We prove that $\mathcal{I}^*$ can be homomorphically mapped to every model $\mathcal{I}$ of $\Gamma' \cup \Delta'$. Since UCQs are basically $\mathbf{Fo}^+_{\exists}$ formulas, which are preserved under homomorphisms (Theorem 3.4.2, Chapter 4), the claim will follow. We can define h as the identity over $\mathit{adom}(\Delta')$. For the other elements we apply the following argument. Notice that for every sentence $\varphi_1, \ldots, \varphi_{\#(\Gamma')} \in \Gamma'$, there might be ≥ 1 elements $c \in \mathbb{D}_{\mathcal{I}}$ bound to the existentially quantified variable y by some assignment γ (i.e., s.t. $\gamma(y) := c$). Let $\mathbf{c}$ be a choice function from $\{1, \ldots, \#(\Gamma)\}$ to $[c_1] \cup \cdots \cup [c_{\#(\Gamma)}]$ (i.e., s.t. $\mathbf{c}(i) \in [c_i]$, for all $i \in [1, \#(\Gamma)]$). Modulo (**AC**), we can assume that $\mathbf{c}$ makes always the "right" choice. Thus, we set $h([c_i]) := \mathbf{c}(i)$, for $i \in [1, \#(\Gamma)]$. This is depicted by Figure 7.2.

Finally, by combining ($\dagger$) with Lemma 7.4.2 we close the proof. □

Theorem 7.4.5. KBQA *is in* **PTime** *in data complexity for IS-A$^{\forall}_{\exists}$+TV and UCQs.*

Proof. Let $\Gamma \cup \Delta$ be a set of IS-A$^{\forall}_{\exists}$+TV meaning representations. Let $\varphi(\bar{x})$ be a UCQ. Let $\bar{c} \in \mathbf{Dom}^{|\bar{x}|}$. Let $\Gamma \cup \Delta$ be as in Lemma 7.4.2 and $\mathcal{I}^*$ as in Lemma 7.4.4 We now show that there exists a a positive **DATALOG** program $\mathcal{P}^{\varphi}_{\Gamma'}$ such that, for all $\bar{c} \in \mathbf{Dom}$, $\bar{c} \in \mathit{cert}(\varphi, \Gamma, \Delta)$ iff $\bar{c} \in \mathcal{P}^{\varphi}_{\Gamma'}(\Delta')$, by showing that $\mathcal{I}^*$ is essentially the minimal model of this program up to homomorphical equivalence.

Notice that $\varphi(\bar{x})$ is already a positive **DATALOG** goal. Notice too that Δ' is a set of positive **DATALOG** ground facts. Thus, we only need to take care of Γ'. Recall that in Γ' there are at most $\#(\Gamma')$ sentences of the form $\varphi_i := \forall x(\varphi(x) \Rightarrow \exists y \psi(x, y))$. Replace each φ_i with $\varphi'_i := \forall x(\varphi(x) \Rightarrow \psi(x, c'_i))$ where c'_i is a fresh Skolem constant, yielding Γ^*. By clausifying Γ^* we obtain a set of positive **DATALOG** rules. Denote by $\mathcal{P}^{\varphi}_{\Gamma'}$ the resulting program. We now claim that

$$\mathcal{I}^* \models \varphi(\bar{c}) \quad \text{iff} \quad \bar{c} \in \mathcal{P}^{\varphi}_{\Gamma'}(\Delta'), \tag{$\dagger$}$$

($\Rightarrow$) Let $\mathcal{H}^*$ be the least Herbrand model of $\mathcal{P}^{\varphi}_{\Gamma'} \cup \Delta'$. Define h as the identity over $\mathit{adom}(\Delta)$ and put, for each $1 \leq i \leq \#(\Gamma')$, $h([c_i]) := c'_i$. Clearly, h is an homomorphism from $\mathcal{I}^*$ to $\mathcal{H}^*$. Therefore $\mathcal{H}^* \models \varphi(\bar{c})$ (since UCQs are closed under homomorphisms) and a fortiori $c \in \mathcal{P}^{\varphi}_{\Gamma'}(\Delta')$, as desired.

($\Leftarrow$) By hypothesis, $\mathcal{H}^* \models \varphi(c)$. Define an homomorphism h from $\mathcal{H}^*$ to $\mathcal{I}^*$ as the identity over $\mathit{adom}(\Delta)$ and by putting, for each $1 \leq i \leq \#(\Gamma')$, $h(c'_i) := [c_i]$. Again, as UCQs are closed under homomorphisms, the claim follows.

Define now a query answering algorithm for IS-A$^{\forall}_{\exists}$+TV and UCQs as follows. Given $\Gamma \cup \Delta$, φ and tuple $\bar{c}$:

1. compute, in time constant in $\#(\Delta)$, $\Gamma' \cup \Delta$,
2. compute, in time constant in $\#(\Delta)$, $\mathcal{P}^{\varphi}_{\Gamma'}$, and
3. check, in time polynomial in $\#(\Delta')$ and a fortiori in $\#(\Delta')$, whether $\bar{c} \in \mathcal{P}^{\varphi}_{\Gamma'}(\Delta')$.

That this algorithm is sound and complete can be proven thus:

$$\begin{array}{lll} \bar{c} \in \mathit{cert}(\varphi, \Gamma, \Delta) & \text{iff} & \mathcal{I}^* \models \varphi(\bar{c}) \quad \text{(by Lemma 7.4.4)} \\ & \text{iff} & \bar{c} \in \mathcal{P}^{\varphi}_{\Gamma'}(\Delta') \quad \text{(by ($\dagger$))} \end{array}$$

Which means that checking whether $\bar{c}$ is an answer can be done in time polynomial on $\#(\Delta)$. This closes the proof. □

Remark 7.4.6. If we replace "every" with "only" in predicate position, then KBQA becomes **NLSpace**-complete for IS-A$^{\forall}_{\exists}$+TV. On the one hand, Calvanese et al. show in [CdL$^+$06] (by reduction from the reachability problem for directed graphs) that any logic capable of expressing **FO** sentences of the form $\forall x(P(x) \Rightarrow \forall y(S(x, y) \Rightarrow Q(y))$, or, equivalently, of the form $\forall x(\exists y(S(y, x) \wedge P(y) \Rightarrow Q(x))$, is **NLSpace**-hard for KBQA. We can express such assertions with sentences of the form "Every P Ss only Qs", whence the lower bound. We recall that meaning representations for "only" (a.k.a. *universal restrictions*) are of the form

$$\lambda P.\lambda Q.\forall x(Q(x) \Rightarrow P(x))$$

(notice that P and Q get inverted). On the other hand, it turns out that all of the meaning representations generated by our fragment would be linear **DATALOG** rules, goals and facts and KBQA for linear **DATALOG** is in **NLSpace** (see [EGDV01], Theorem 4.3).

Corollary 7.4.7. *The combined complexity of* KBQA *for IS-A$^{\forall}_{\exists}$+TV meaning representations and (U)CQs* **NPTime**-*complete.*

7.5 The Fragments of English.

We now turn to Pratt and Third's fragments of English. In [PHT06] they show how to design resolution procedures to decide SAT (see also Table 2.2 in Chapter 4), which imply as corrollary, tight complexity bounds for the combined complexity of KBSAT and lower bounds for, again the combined complexity of KBQA. In this section we complement these results with data complexity results for KBSAT and KBQA and with complexity upper bounds for the combined complexity of KBQA. In Figure 7.3 the relative expressive power of the fragments is spelled out. Note also that, by definition, the positive fragment IS-$A^{\forall}_{\exists}$ is subsumed by COP, IS-$A^{\forall}_{\exists}$+TV by COP+TV, IS-$A^{\forall}_{\exists}$+DTV by COP+DTV and IS-$A^{\forall}_{\exists}$+TV+DTV and by COP+TV+DTV. Unrestricted resolution does not terminate on even some of the simplest fragments of English.

Proposition 7.5.1. *Unrestricted resolution does not terminate on COP+TV meaning representations.*

Proof. Consider the COP+TV sentences "Every man trusts some male.", "Every male is a man." and "John is a male.", their meaning representations and the clauses thereof derivable:

$$\Gamma = \left\{ \begin{array}{c} \overline{\mathit{Man}(x)} \vee \mathit{Trusts}(x, f(x)), \overline{\mathit{Man}(x)} \vee \mathit{Male}(f(x)), \\ \overline{\mathit{Male}(x)} \vee \mathit{Man}(x), \mathit{Man}(\text{John}) \end{array} \right\}.$$

Then Γ^{∞} is infinite, since $\{\mathit{Man}(f^i(\text{John})) \mid i \in \mathbb{N}\} \subseteq \Gamma^{\infty}$. □

However, an important Lemma by Pratt and Third in [PHT06] that we state below, allows us to reduce SAT to SAT for *unary* clauses from $\mathbf{SC}^+$ and apply the (terminating) resolution procedures from Table 7.1 to derive complexity upper bounds.

Lemma 7.5.2 ([PHT06], Lemma 4.5)**.** *Let Γ be a set of clauses obtained by Skolemization and clausification from COP+TV+DTV+Rel meaning representations (or any fragment thereof). Then, we can construct in time polynomial on $\#(\Gamma)$, a set Γ_u of unary clauses (of depth $d(\Gamma_u) \geq d(\Gamma)$) s.t. Γ is satisfiable iff Γ_u is satisfiable.*

For studying in particular KBQA for the positive fragments and the fragments of English, we consider in this section three classes of queries:

- UCQs,
- TCQs and
- generalized tree shaped queries (GTCQs)

where GTCQs are defined thus:

Definition 7.5.3. A *generalized tree-shaped conjunctive query* (GTCQ) $\varphi(x)$ is a query of arity $n = 1$ defined as follows:

$$\begin{array}{rl} \varphi(x) \rightarrow & P(x) \mid \exists y S(x,y) \mid \exists y \exists z T(x,y,z) \mid \varphi'(x) \wedge \varphi''(x) \\ & \mid \exists y (S(x,y) \wedge \varphi(y)) \mid \exists x \exists z (T(x,y,z) \wedge \varphi'(y) \wedge \varphi''(z)) \end{array} \qquad \text{(GTCQ)}$$

which is basically the generalization of TCQs to signatures $\mathbf{Sig} = (\emptyset, \emptyset, \mathbf{R})$ where $\mathbf{R}$ contains both binary and ternary predicate symbols.

7.5.1 Fragments with Tractable Data Complexity

Theorem 7.5.4. KBQA *is in* **PTime** *in data complexity for COP+TV meaning representations and UCQs.*

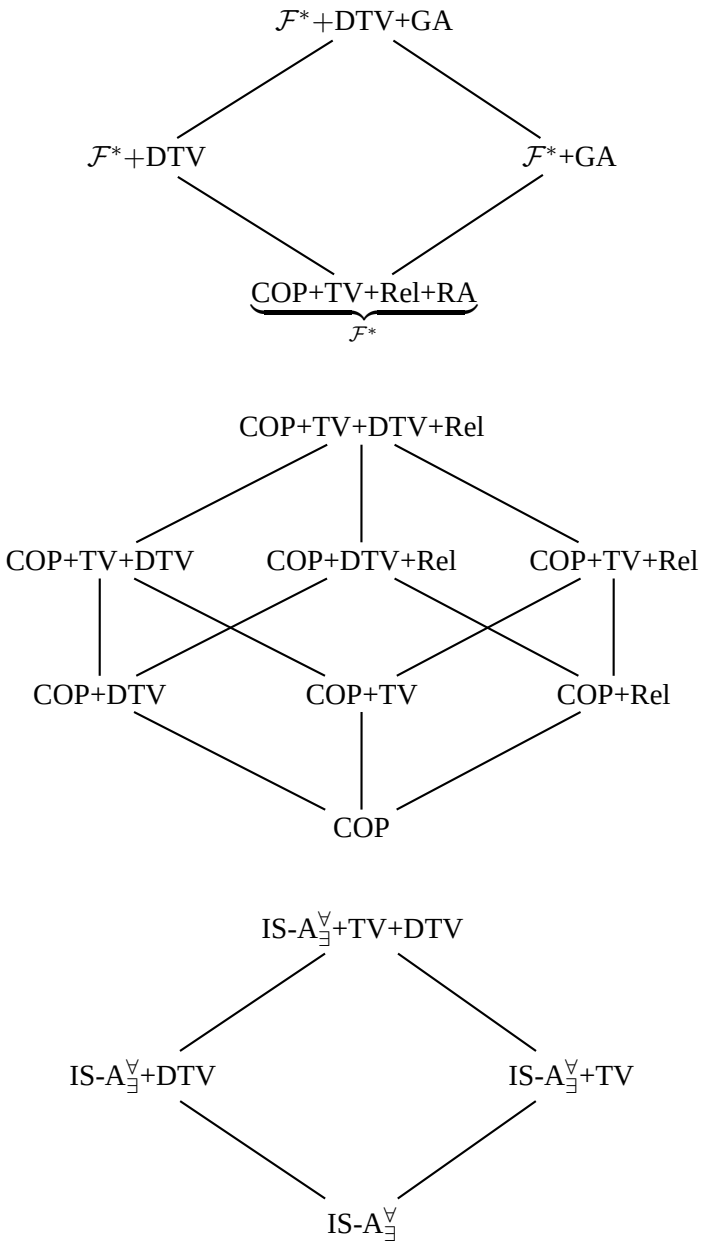


Figure 7.3: Comparative expressive power of the fragments of English. The top diagram shows the *minimal* undecidable fragments [PHT06].

Proof. Let $\Gamma \cup \Delta$ be a set of COP+TV meaning representations, φ a UCQ of arity n and body $\varphi(\bar{x})$, and $\bar{c}$ be a sequence of n constants. Consider Γ. We say that a sentence $\varphi \in \Gamma$ is *positive*, denoted φ^+, if φ is of the form

$$\forall x(\varphi(x) \Rightarrow \psi(x)), \quad \forall x(\varphi(x) \Rightarrow \forall y\psi(x,y)), \quad \text{or} \quad \forall x(\varphi(x) \Rightarrow \exists y\psi(x,y))$$

We say that φ is *negative*, denoted φ^-, when φ is of the form

$$\forall x(\varphi(x) \Rightarrow \neg\psi(x)), \quad \forall x(\varphi(x) \Rightarrow \neg\forall y\psi(x,y)), \quad \text{or} \quad \forall x(\varphi(x) \Rightarrow \neg\exists y\psi(x,y)).$$

The facts in Δ are treated similarly. This induces a partitioning of Γ into $\Gamma^+ \cup \Gamma^-$ and of Δ into $\Delta^+ \cup \Delta^-$. We claim that, if $\Gamma \cup \Delta$ is satisfiable, then it holds that

$$\bar{c} \in \mathit{cert}(\varphi, \Gamma, \Delta) \quad \text{iff} \quad \bar{c} \in \mathit{cert}(q, \Gamma^+, \Delta^+). \tag{$\dagger$}$$

Assume that $\Gamma \cup \Delta$ is satisfiable. The ($\Leftarrow$) direction is immediate, since $\Gamma^+ \cup \Delta^+ \subseteq \Gamma \cup \Delta$. To prove the converse, suppose for contradiction that $\bar{c} \notin \mathit{cert}(\varphi, \Gamma^+, \Delta^+)$. Then, there exists a model $\mathcal{I}$ of $\Gamma^+ \cup \Delta^+$ s.t. $\mathcal{I} \not\models \varphi(\bar{c})$. On the other hand, since by assumption, $\Gamma \cup \Delta$ is satisfiable and $\bar{c} \in \mathit{cert}(\varphi, \Gamma, \Delta)$, then for some model $\mathcal{I}'$ of $\Gamma \cup \Delta$, $\mathcal{I}' \models \varphi(\bar{c})$. Define an homomorphism h from $\mathcal{I}'$ to $\mathcal{I}$ by putting:

- on the one hand, $h(c) := c$, for all $c \in \mathit{adom}(\Delta)$ (notice that $\mathit{adom}(\Delta)$ is a subset of both $\mathbb{D}_{\mathcal{I}}$ and $\mathbb{D}_{\mathcal{I}'}$), and
- for all $c_1, c_2 \in \mathbb{D}_{\mathcal{I}'} \setminus \mathit{adom}(\Delta)$, if $(c_1, c_2) \in R^{\mathcal{I}'}$, $h(c_i) :=$ the least (w.r.t. lexicographic ordering) $c_i' \in \mathbb{D}_{\mathcal{I}'}$, for $i = 1, 2$, s.t. $(c_1', c_2') \in R^{\mathcal{I}}$.

Clearly $\mathcal{I}' \hookrightarrow_h \mathcal{I}$, whence, since UCQs are closed under homomorphisms (Theorem 3.4.2), it follows that $\mathcal{I} \models \varphi(\bar{c})$. Contradiction.

Notice that $\Gamma^+ \cup \Delta^+$ is now a set of IS-A$^{\forall}_{\exists}$+TV meaning representations to which we can apply the (query answering) algorithm sketched in Theorem 7.4.5. Hence, sketch a query answering algorithm for COP+TV as follows. Given $\Gamma \cup \Delta$, φ and sequence $\bar{c}$:

1. check, in time polynomial in $\#(\Delta)$, whether $\Gamma \cup \Delta$ is satisfiable,
2. if it is unsatisfiable, answer yes,
3. otherwise, check, in time polynomial in $\#(\Delta^+)$ and a fortiori in $\#(\Delta)$, whether it is the case that $\bar{c} \in \mathit{cert}(\varphi, \Gamma^+, \Delta^+)$.

This algorithm (on the grounds of ($\dagger$)) is trivially sound and complete and runs in time polynomial in $\#(\Delta)$. □

The theorem sketches also an **NPTime** algorithm in combined complexity. Since KBQA is bounded below by SAT in combined complexity, it follows from Table 2.2, Chapter 2, that:

Corollary 7.5.5. KBQA *is* **NPTime***-complete in combined complexity for COP+TV and (U)CQs*

Theorem 7.5.6. KBSAT *is in* **LSpace** *in data complexity for*

1. *COP+TV and*
2. *COP+TV+DTV.*

Proof. Consider a set $\Gamma \cup \Delta$ of COP+TV+DTV and/or COP+TV meaning representations (with no cycles). Since we are interested only in data complexity, we can assume Γ to be fixed. Let Γ^{cl} be the *(i)* Skolemization and *(ii)* clausification of Γ (this can be done in constant time, since Γ is

fixed). Clearly, $\Delta^{cl} = \Delta$. Pratt and Third in [PHT06] show that the clauses in $\Gamma^{cl} \cup \Delta^{cl}$ are of the form

$$\left.\begin{array}{ll} \pm P(c) & \pm L \\ \neg P(x) \vee \pm Q(x) & \neg P(x) \vee \pm L(x) \\ \neg P(x) \vee Q(f(x)) & \neg P(x) \vee \neg Q(y) \vee \pm L(x,y) \\ \neg P(x) \vee \neg Q(y) \vee \neg N(z) \pm L(x,y,z) & \\ \neg P(x) \vee \neg Q(y) \vee N(g(x,y)) & \end{array}\right\} = \text{COP+TV} \left.\right\} = \text{COP+TV+DTV}$$

where $L(\bar{x})$ denotes a unary, binary or ternary literal over the variables $\bar{x}$: COP+TV meaning representations use unary and binary relation symbols, whereas COP+TV+DTV meaning representations make use of unary, binary and ternary relation symbols. For example, L, which contains no free variables, can be of the form $P(c)$, $R(c, c')$, $P(f(c))$, etc.

We can assume w.l.o.g. Δ^{cl} to contain only positive ground atoms, since, similarly to the proof of Theorem 4.1.15 from Chapter 5, we can "define out" a negative binary ground atom (unary atoms are dealt with similarly) $\neg R(c, c')$ by introducing a fresh relation S, a *disjointness* rule $\neg R(x, y) \vee \neg S(x, y)$ and, finally, *(i)* replacing every occurrence of R by S in Γ^{cl} and *(ii)* replacing the atom $\neg R(c, c')$ by $S(c, c')$. This transformation does not affect data complexity, since it does not affect tuples of constants.

To prove claim **(1)** we reason as follows. Observe that COP+TV clauses satisfy the following properties:

- all literals are covering and
- literals are either monadic or, if L is not monadic in clause C, then $\mathit{Var}(C) = \mathit{Var}(L)$,

which imply that $\Gamma^{cl} \cup \Delta^{cl} \subseteq \mathbf{SC}^+$. In addition, by inspection, we can see that applying the *res*, *fact* and *cond* rules results in *(i)* clause length not growing beyond some finite bound $l \in \mathbb{N}$ and *(ii)* covering clauses. Therefore, the A-ordered resolution calculus $\mathcal{R}_{2,4}$ from Table 7.1 is a decision procedure for such clauses, since $\prec_d$ prevents clause depth from growing beyond some finite bound $d \in \mathbb{N}$. Moreover, since ground atoms $P(c), R(c, c') \in \Delta^{cl}$ are of null depth (and so are the ground atoms generated when resolving clauses from solely Γ^{cl}), we can apply the "separation" Lemma 7.1.6 and saturate first Γ^{cl}, yielding the finite saturation (that does not depend on the data) $(\Gamma^{cl})^{\infty}$.

Inconsistency in $(\Gamma^{cl})^{\infty} \cup \Delta^{cl}$ may arise by *(i)* conflicting constraints in $(\Gamma^{cl})^{\infty}$ or by *(ii)* the data in Δ^{cl} being in conflict with the constraints in $(\Gamma^{cl})^{\infty}$, in which case $\bot$ would be derived in at most $\mathbf{O}(\#(\Delta))$ steps. Now, checking whether $(\Gamma^{cl})^{\infty}$ is independant from the data in Δ^{cl} and can be done in time constant in $\#(\Delta)$. In addition to this, it is clear that Δ^{cl} can be saved in a register using at most $\mathbf{O}(\mathtt{log}\,\#(\Delta))$ space. Hence, an algorithm checking for the satisfiability of $((\Gamma^{cl})^{\infty} \cup \Delta)^{\infty}$, and hence of $\Gamma \cup \Delta$, would proceed by looping $\mathbf{O}(\#(\Delta))$ times over Δ^{cl} until either $\bot$ is derived or no other derivation steps are possible.

The proof of claim **(2)** is similar. COP+TV+DTV clauses satisfy the following property:

- literals are either monadic or, if L is not monadic in clause C, then $\mathit{Var}(C) = \mathit{Var}(L)$

although they may not be covering, i.e., there might be some clause C with a literal $\pm L(x, y) := \pm L'(x, y, f(x))$ which is *not* covering (even though $\mathit{Var}(C) = \mathit{Var}(L')$. However, they are still contained in $\mathbf{SC}^+$ and A-ordered resolution can be used to decide KBSAT.

Indeed, literal $L'(x, y, f(x))$ is *almost monadic* on x, hence, applying (**Mon**) would yield the covering literal $L'(x, x, f(x))$ (recall that by Theorem 7.2.7, adding (**Mon**) to $\mathcal{R}_d$ with *cond* and, possibly, *split*, gives rise to sound and refutation-complete calculi for $\mathbf{SC}^+$). On the other hand, by inspection we can see that resolution does not make the length of clauses increase beyond a finite

bound $l \in \mathbb{N}$. To control the depth d we use the A-ordered resolution calculus $\mathcal{R}_{2,5}$. Thereafter we reason as before, applying Lemma 7.1.6. □

Corollary 7.5.7. *The data complexity of* KBSAT *is in* **LSpace** *for IS-A$^{\forall}_{\exists}$+TV, IS-A$^{\forall}_{\exists}$+DTV and IS-A$^{\forall}_{\exists}$+TV+DTV.*

Theorem 7.5.8. *The data complexity of* KBSAT *is in* **LSpace** *for COP+Rel.*

Proof. Let $\Sigma \cup \Delta$ be a set of COP+Rel meaning representations, where Σ is fixed (and $\#(\Sigma)$ a constant) and let $\Sigma^{cl} \cup \Delta^{cl}$ be their clausification and Skolemization. $\Sigma^{cl} \cup \Delta^{cl}$ can be computed in $\mathbf{O}(\mathtt{log}\,\#(\mathit{\Delta}))$ space. This gives way to $\#(\Delta) + k$ constants, for some fixed integer $k \leq \#(\Sigma)$. Now, the clauses in $\Sigma^{cl} \cup \Delta^{cl}$ are

- monadic, and
- if in Σ^{cl}, Boolean combinations of unary atoms over the single variable x, and containing no function symbols.

Thus, satisfiability (by the Herbrand theorem) reduces to Herbrand satisfiability and propositional satisfiability, by computing $\mathit{GR}(\Sigma^{cl}, \Delta^{cl})$, i.e., the set of propositional clauses resulting from grounding the clauses in Σ^{cl} with the constants $c \in \mathit{adom}(\Delta^{cl})$[1]. Since Σ is fixed, the $\#(\Sigma^{cl}) \cdot \#(\Delta^{cl})$ groundings can be stored using $\mathbf{O}(\mathtt{log}\,\#(\mathit{\Delta}))$ space.

Let p be the number of unary predicates occurring among the clauses in Σ^{cl}; as such it is a constant that depends on Σ^{cl}. Since the Herbrand domain $\mathit{HD}_{\Sigma^{cl} \cup \Delta^{cl}}$ of $\Sigma^{cl} \cup \Delta^{cl}$ is $\mathit{adom}(\Delta) \cup \{c_1, \ldots, c_k\}$, the Herbrand base $\mathit{HB}_{\Sigma^{cl} \cup \Delta^{cl}}$ is of size $p \cdot \#(\Delta) + k$ and can be stored, again, using $\mathbf{O}(\mathtt{log}\,\#(\mathit{\Delta}))$ space.

The models, that is, the truth value assignments $\delta(\cdot)$ for $\mathit{GR}(\Sigma^{cl}, \Delta^{cl})$ are essentially the Herbrand models $\mathcal{H} \subseteq \mathit{HB}_{\Sigma^{cl} \cup \Delta^{cl}}$ of $\Sigma^{cl} \cup \Delta^{cl}$, since

$$\delta(p_{P(c)}) = 1 \quad \text{iff} \quad P(c) \in \mathcal{H},$$

of which $\leq 2^{p \cdot \#(\Delta) + k}$ (i.e., finitely many) of size $\leq p \cdot \#(\Delta) + k$ exist, each of which can be stored using at most $\mathbf{O}(\mathtt{log}\,\#(\mathit{\Delta}))$ space.

A satisfiability checking algorithm will loop through such space of $\leq 2^{p \cdot \#(\Delta) + k}$ truth value assigments until some $\mathcal{H} \supseteq \Delta^{cl}$ s.t. $\mathcal{H} \models \Sigma^{cl}$ is found, in which case it would return "true", or else, if no such model exists, it will return "false" after its last iteration. As it would at most $\mathbf{O}(\mathtt{log}\,\#(\mathit{\Delta}))$ space, the result follows. □

7.5.2 Fragments with Intractable Data Complexity

Lemma 7.5.9. KBQA *is* **coNPTime**-*hard in data complexity for COP+Rel and TCQs.*

Proof. We define a reduction from 2+2-SAT (recall Chapter 6, Section 5.5). Let $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ be a 2+2-formula over the propositional atoms $\mathit{At}(\psi) := \{l_1, ..., l_m\}$ where, for $i \in [1, k]$,

$$\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

is a disjunction of two non-negated and two negated propositional atoms from $\{l_1, \ldots, l_m\}$.

To encode ψ, we give ourselves four transitive verbs, N_1 (for "has as first negative atom"), N_2 (for "has as second negative atom"), P_1 (for "has as first positive atom") and P_2 (for "has as second negative atom"), plus the nouns A (for "atom"), A_f (for "false atom") and A_t (for "true atom"). For each conjunct ψ_i in ψ, for $i \in [1, k]$, we introduce a proper name c_i and a proper name l for every $l \in \mathit{At}(\psi)$.

We encode ψ into

[1] The ground atom $P(c)$ is encoded by the propositional atom $p_{P(c)}$.

$$\mathcal{F}_\psi := \left\{ \begin{array}{l} p_{11} \text{ is an } A.\ p_{12} \text{ is an } A.\ n_{11} \text{ is an } A.\ n_{12} \text{ is an } A. \\ \vdots \\ p_{k1} \text{ is an } A.\ p_{k2} \text{ is an } A.\ n_{k1} \text{ is an } A.\ n_{k2} \text{ is an } A. \\ c_1\ P_1\text{s}\ p_{11}.\ c_1\ P_2\text{s}\ p_{12}.\ c_1\ N_1\text{s}\ n_{11}.\ c_1\ N_2\text{s}\ n_{12}. \\ \vdots \\ c_k\ P_1\text{s}\ p_{k1}.\ c_k\ P_2\text{s}\ p_{k2}.\ c_k\ N_1\text{s}\ n_{k1}.\ c_k\ N_2\text{s}\ n_{k2}. \end{array} \right\}$$

and consider, resp., the sentences and the TCQ

$$\mathcal{S} := \left\{ \begin{array}{l} \text{No } A \text{ is not an } A_t \text{ that is not an } A_f. \\ \text{No } A_t \text{ is an } A_f.\ \text{No } A_f \text{ is an } A_t. \end{array} \right\}$$

$$\begin{array}{rcl} \varphi & := & \exists x \exists y (P_1(x,y) \wedge A_f(y)) \wedge \exists z (P_2(x,z) \wedge A_f(z)) \wedge \\ & & \exists w (N_1(x,w) \wedge A_t(w)) \wedge \exists v (N_2(x,v) \wedge A_t(v))). \end{array}$$

Observe that φ could have been expressed by the GCQ-English question Q_ψ: "Does somebody P_1 some A_f and P_2 some A_f and N_1 some A_t and N_2 some A_t?". We now claim that

$$\psi \text{ is satisfiable} \qquad \text{iff} \qquad \tau(\mathcal{S} \cup \mathcal{F}_\psi) \not\models \varphi \tag{$\dagger$}$$

($\Rightarrow$) Suppose that ψ is satisfiable and let $\delta(\cdot)$ be such model (a truth assignment). Then, for all $i \in [1,k]$, $\delta(\psi_i) = 1$, i.e., $\delta(p_{i1}) = 1$ or $\delta(p_{i2}) = 1$ or $\delta(n_{i1}) = 0$ or $\delta(n_{i2}) = 0$. Given this, we can construct an interpretation $\mathcal{I} = (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ s.t. $\mathcal{I} \models \tau(\mathcal{S} \cup \mathcal{F}_\psi)$ but $\mathcal{I} \not\models \varphi$ as follows:

- $\mathbb{D}_\mathcal{I} := \{c_i, p_{ij}, n_{ij} \mid i \in [1,k], j = 1,2\}$,
- $A^\mathcal{I} := \{l \in \mathit{At}(\psi) \mid A(l) \in \tau(\mathcal{F}_\psi)\}$,
- $P_j^\mathcal{I} := \{(c_i, p_{ij}) \mid P_j(c_i, p_{ij}) \in \tau(\mathcal{F}_\psi), i \in [1,k]\}$,
- $N_j^\mathcal{I} := \{(c_i, n_{ij}) \mid N_j(c_i, n_{ij}) \in \tau(\mathcal{F}_\psi), i \in [1,k]\}$,
- $A_f^\mathcal{I} := \{l \in A^\mathcal{I} \mid \delta(l) = 0\}$ and
- $A_t^\mathcal{I} := \{l \in A^\mathcal{I} \mid \delta(l) = 1\}$.

($\Leftarrow$) Let $\mathcal{I}$ be a model $\tau(\mathcal{S} \cup \mathcal{F}_\psi)$ of s.t. $\mathcal{I}, \gamma \not\models \varphi$ for all γ. We want to show that there exists a ta $\delta(\cdot)$ s.t. $\delta(\psi) = 1$. Let $\delta \colon \mathit{At}(\psi) \to \{0,1\}$ be the truth assignment s.t.

$$\delta(l) = 1 \quad \text{iff} \quad l \in A_t^\mathcal{I}.$$

Now, by assumption $\mathcal{I}, \gamma \not\models \varphi$, for all γ. This implies, for all $i \in [1,k]$, that either $p_{i1} \notin A_f^\mathcal{I}$ or $p_{i2} \notin A_f^\mathcal{I}$ or $n_{i1} \notin A_t^\mathcal{I}$ or $n_{i2} \notin A_t^\mathcal{I}$. Now, recall that $\mathcal{I} \models \tau(\mathcal{S})$, where $\tau(\mathcal{S})$ contains the axioms

$$\forall x (A(x) \Rightarrow A_t(x) \vee A_f(x)), \quad \forall x (A_t(x) \Rightarrow \neg A_f(x)), \quad \forall x (A_f(x) \Rightarrow \neg A_t(x))$$

that "say" that an atom is either true or false, but not both. Hence if $p_{i1} \notin A_f^\mathcal{I}$, then, by definition of $\delta(\cdot)$, $\delta(p_{i1}) = 1$ and similarly for the other cases. Therefore, $\delta(\psi_i) = 1$, for all $i \in [1,k]$, and thus $\delta(\psi) = 1$. □

Lemma 7.5.10. KBQA *is* **coNPTime***-hard in data complexity for COP+Rel+DTV and GTCQs.*

Proof. The **coNPTime** data complexity lower bound for COP+Rel+DTV follows also by reduction from 2+2-SAT, by considering the following minor adjustments to the reduction defined in Lemma 7.5.9. Leave $\mathcal{S}$ unchanged. Regarding φ and $\mathcal{F}_\psi$ proceed as follows. Instead of considering N_j and P_j, for $j = 1,2$, binary, we consider them $\mathbf{DTV}$s expressing *ternary* predicate symbols. As such, whenever, for $i \in [1,k]$, $p_{i1} \vee p_{i_2} \vee \neg n_{i1} \vee \neg n_{i2}$ is one of the k conjuncts ψ_i of

a 2+2 formula ψ over the propsitional atoms $\{l_1, \ldots, l_m\}$, add the facts "c_i P_1s p_{i1} to c_i", "c_i P_2s p_{i2} to c", "c_i N_1s n_{i1} to c" and "c_i N_2s n_{i1} to c" to $\mathcal{F}_\psi$, where c is a "dummy" $\mathbf{Pn}$ different from every other $\mathbf{Pn}$ in $\mathit{adom}(\mathcal{F})$. Finally, consider the GTCQ

$$\begin{aligned}\varphi \quad := \quad & \exists x \exists y (P_1(x, y, c) \wedge A_f(y)) \wedge \exists z (P_2(x, z, c) \wedge A_f(z)) \wedge \\ & \exists w (N_1(x, w, c) \wedge A_t(w)) \wedge \exists v (N_2(x, v, c) \wedge A_t(v))).\end{aligned}$$

The remainder of the proof is analogous to that for COP+Rel and TCQs. □

Lemma 7.5.11. KBQA *is in* **coNPTime** *in data complexity for COP+Rel+TV+DTV and GTCQs.*

Proof. Observe that GTCQs φ are expressible by COP+Rel+TV+DTV meaning representations (it is trivial to extend its grammar to express such formal queries). Also, the negation $\neg\varphi$ of a GCTQ φ is expressible in COP+Rel+TV+DTV. Thus, given a set $\Gamma \cup \Delta$ of COP+Rel+TV+DTV meaning representations and a GTCQ φ, it is immediate to see that

$$\Gamma \cup \Delta \models \varphi \quad \text{iff} \quad \Gamma \cup \Delta \cup \{\neg\varphi\} \text{ is unsatisfiable.} \qquad (\dagger)$$

Moreover, such a reduction is constant in $\#(\Delta)$. If we where able to reduce $\Gamma \cup \Delta \cup \{\neg\varphi\}$ to a set of $\mathbf{SC}^+$ clauses and apply Theorem 7.3.2, we would derive a **coNPTime** data complexity (upper) bound for answering (G)TCQs.

Pratt and Third in [PHT06] show that COP+Rel+TV+DTV meaning representations in *clausal form* have one of the following forms:

$\pm P(c)$	$\pm L$
$\neg P(x) \vee \pm Q(x)$	$\neg P(x) \vee \pm L(x)$
$\neg P(x) \vee \pm Q(x) \vee N(x)$	$\neg P(x) \vee \neg Q(y) \vee \pm L(x, y)$
$\neg P(x) \vee Q(f(x))$	$\neg P(x) \vee \neg Q(y) \vee \neg N(z) \vee \pm L(x, y, z)$
	$\neg P(x) \vee \neg Q(y) \vee N(g(x, y))$

where $L(\bar{x})$ denotes as before a unary, binary or ternary literal over the variables $\bar{x}$.

By applying Pratt and Third's Lemma 7.5.2, $\Gamma \cup \Delta \cup \{\neg\varphi\}$ can be transformed into a set of *unary* clauses $K_u^{cl} := \Gamma_u^{cl} \cup \Delta_u^{cl} \cup \{\overline{\varphi_u^{cl}}\}$. By inspection, one can see that, for all clauses C in K_u^{cl} and all literals $L \in \mathit{Lit}(C)$,

- either $\#(\mathit{Var}(L)) \leq 1$ or
- $\mathit{Var}(L) = \mathit{Var}(C)$.

Thus, K_u^{cl} is a set of $\mathbf{SC}^+$ clauses. Thereafter, A-ordered resolution with monadization and splitting (e.g., $\mathcal{R}_{3,8}$) can be used to guess a saturation K_u^{cl}, which we can then ground in time polynomial in, ultimately, $\#(\Delta)$ and check for satisfiability, by guessing a truth assignment $\delta(\cdot)$, again, in time polynomial in $\#(\Delta)$. Hence, KBQA is in **coNPTime**. □

Theorem 7.5.12. *The data complexity of* KBQA *is* **coNPTime***-complete for the following fragments: COP+Rel, COP+Rel+DTV, COP+Rel+DTV and COP+Rel+TV+DTV.*

Proof. The hardness follows from Lemma 7.5.9. Membership follows from Lemma 7.5.11. □

Theorem 7.5.13. *The data complexity of* KBSAT *is* **NPTime***-complete for the following fragments: COP+Rel+TV, COP+Rel+DTV and COP+Rel+TV+DTV.*

Proof. Membership in **NPTime** for COP+Rel+TV and COP+Rel+TV+DTV is derived as follows. Consider a set $\Gamma \cup \Delta$ of COP+Rel+TV or COP+Rel+TV+DTV meaning representations. Clausifying such meaning representations can be done in time constant in $\#(\Delta)$. By Lemma 7.5.2, we know that we can reduce, in time polynomial in $\#(\Delta)$, their satisfiability to that of a set $\Gamma_u \cup \Delta_u$

of monadic clauses. By inspection we can moreover observe that such classes belong to the $\mathbf{Sc}^{+}$ class. We can now apply Theorem 7.3.2, whence it follows that KBSAT is in **NPTime**.

On the other hand, **NPTime**-hardness for the fragments COP+Rel+TV, COP+Rel+DTV and COP+Rel+TV+DTV can be inferred by a reduction from the **NPTime**-complete 2+2-SAT problem (by means of proofs analogous to those for KBQA). □

7.5.3 Combined Complexity

Theorem 7.5.14. *The combined complexity of* KBQA *for arbitrary CQs is* **NPTime***-hard for the six fragments IS-A$^{\forall}_{\exists}$+TV, IS-A$^{\forall}_{\exists}$+DTV, IS-A$^{\forall}_{\exists}$+TV+DTV, COP+TV, COP+DTV and COP+TV+DTV.*

Proof. Checking whether a tuple $\bar{c}$ is an answer to CQ φ over an arbitrary database $\mathcal{D}$/set $\mathcal{F}$ of facts is **NPTime**-hard in $\mathcal{D}/\mathcal{F}$ and φ (with $\mathcal{O}/\mathcal{S}$ *empty*), by reduction from, e.g., the **NPTime**-complete graph homomorphism problem (recall the proof of Theorem 4.3.1 from Chapter 5). This lower bound propagates to all the fragments. □

Theorem 7.5.15 ([PHT06, PH08b, PHM09, PH09])**.** *The combined complexity of* KBSAT *is*

1. *in* **NLSpace** *for IS-A$^{\forall}_{\exists}$ and COP,*
2. **NLSpace***-complete for IS-A$^{\forall}_{\exists}$+TV and COP+TV,*
3. *in* **PTime** *for the fragments IS-A$^{\forall}_{\exists}$+DTV, IS-A$^{\forall}_{\exists}$+TV+DTV, COP+DTV and COP+TV+DTV,*
4. **NPTime***-complete for COP+Rel,*
5. **ExpTime***-complete for COP+Rel+TV, and*
6. **NExpTime***-complete for COP+Rel+DTV+TV*

Proof. The proof is in all cases immediate: KBSAT for all those fragments is polynomially equivalent in combined complexity to SAT, whose computational complexity for the fragments considered is well-known.

The **NLSpace** upper bound for IS-A$^{\forall}_{\exists}$, IS-A$^{\forall}_{\exists}$+TV, COP and COP+TV follows from [PHT06]. See Table 2.2, Chapter 4.

The **NLSpace** lower bound for IS-A$^{\forall}_{\exists}$+TV and COP+TV follows from [PH08b], Theorem 4.11, where a reduction from the **NLSpace**-complete reachability problem for directed graphs is sketched. However, since the instance (i.e., the directed graph) is encoded there without making use of COP+TV sentences with negations, the reduction holds also for IS-A$^{\forall}_{\exists}$+TV.

The fragments IS-A$^{\forall}_{\exists}$+DTV, IS-A$^{\forall}_{\exists}$+TV+DTV, COP+DTV are all contained in COP+TV+DTV. Since [PHT06] shows that SAT is in **PTime** for COP+TV+DTV, the result follows again from Table 2.2.

Finally, the computational properties for COP+Rel, COP+Rel+TV and COP+Rel+DTV+TV are, yet again, trivial corollaries of Table 2.2. □

Theorem 7.5.16. *The combined complexity of* KBQA *when we consider TCQs is*

1. **coNPTime***-hard for COP+Rel,*
2. **ExpTime***-complete for COP+Rel+TV, and*
3. **coNExpTime***-hard COP+Rel+DTV and COP+Rel+DTV+TV.*

Proof. We can reduce SAT for all these fragments (see, again, Table 2.2 from Chapter 4) to KBQA by reusing the reduction defined in the proof of Theorem 7.5.17. This, added to the observation that **coExpTime** = **ExpTime** (i.e., this decision class is closed under complement) implies the lower bounds.

The membership in **ExpTime** for COP+Rel+TV follows from the fact that *(i)* TCQs can be expressed by COP+Rel+TV by some minor adjustments to its grammar, extending its coverage to questions, and *(ii)* by observing that COP+Rel+TV is closed under negation (see Table 2.1 from Chapter 4). Let $\Gamma \cup \Delta$ be a set of COP+Rel+TV meaning representations, c a constant and $\varphi(x)$ a TCQ. Then, clearly, $\Gamma \cup \Delta \models \varphi(c)$ iff $\Gamma \cup \Delta \cup \{\neg\varphi(c)\}$ is unsatisfiable. Since (un)satisfiability for COP+Rel+TV is in **ExpTime**, we conclude. □

7.5.4 Undecidable Fragments

Proposition 7.5.17. KBQA *is undecidable for COP+Rel+TV+GA, COP+Rel+TV+DTV+GA and COP+Rel+TV+DTV+RA and atomic queries.*

Proof. We reduce the satisfiability problem for these fragments to KBQA's complement. We will consider only the case for COP+Rel+TV+GA, since the other are analogous. Let $\Gamma \cup \Delta$ be a set of COP+Rel+TV+GA meaning representations. Consider now the two COP+Rel+TV+GA sentences, *(i)* "No P is a Q." and *(ii)* "c is an P.", and the (Boolean) atomic query $Q(c)$. Then, for all groundings σ,

$$\Gamma \cup \Delta \cup \{\tau(\text{No } P \text{ is a } Q.), \tau(c \text{ is an } P.)\} \not\models \varphi\sigma \quad \text{iff} \quad \Gamma \cup \Delta \text{ is satisfiable.} \qquad (\dagger)$$

($\Leftarrow$) If $\Gamma \cup \Delta$ is satisfiable, then there exists and interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \Gamma \cup \Delta$. Suppose for contradiction that $\Gamma \cup \Delta \cup \{\tau(\text{No } A \text{ is a } B.), \tau(c \text{ is an } A.)\} \models \varphi\sigma$, i.e., that it holds that $\Gamma \cup \Delta \cup \{\forall x(A(x) \Rightarrow \neg B(x)), A(c)\} \models B(c)$. Then $\mathcal{I} \models B(c)$ and $\mathcal{I} \not\models B(c)$, which is absurd.

($\Rightarrow$) If $\Gamma \cup \Delta \cup \{\tau(\text{No } P \text{ is a } Q.), \tau(c \text{ is an } P.)\} \not\models \varphi\sigma$, then there exists an interpretation $\mathcal{I}$ s.t. $\mathcal{I} \models \Gamma \cup \Delta \cup \{\tau(\text{No } P \text{ is a } Q.), \tau(c \text{ is an } P.)\}$ and $\mathcal{I} \not\models B(c)$. This implies that $\mathcal{I} \models \Gamma \cup \Delta$, for some interpretation $\mathcal{I}$. Therefore, $\Gamma \cup \Delta$ is satisfiable.

Since by Table 2.2 from Chapter 5 we know that COP+Rel+TV+GA is undecidable for SAT, this closes the proof. □

7.5.5 Enriching the Interrogative and Declarative Fragments

Tilings are a family of combinatorial problems which have been widely used to prove the undecidability of several fragments of **FO** and of a wide variety of decision problems. A *tiling grid* or *grid* is a tuple $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$, where $\mathbb{T} := \{c_1, \ldots, c_k\}$ is a finite set of k *tiles* and $\mathbb{V}$ and $\mathbb{H}$ are binary relations over $\mathbb{T}$, called, resp., the *vertical* and *horizontal* relations. A *tiling* is a function $t\colon \mathbb{N} \times \mathbb{N} \to \mathbb{T}$ that verifies the horizontal and vertical constraints, i.e., for all $i, j \in \mathbb{N}$, $(t(i,j), t(i,j+1)) \in \mathbb{H}$ and $(t(i,j), t(i+1,j)) \in \mathbb{V}$. See [GGB01], Appendix A, for a general introduction to tilings.

Definition 7.5.18. The *unbounded tiling problem* (TP) is the undecidable decision problem defined by

- **Input:** a grid $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$.
- **Question:** does a tiling t exist for $\mathcal{T}$?

Ian Pratt in [PH08b] shows that KBQA for $\mathbf{FO}^2$ and CQs is undecidable by a reduction from TP. The same technique is used in [PHT06] to shown that SAT for COP+Rel+TV+GA is undecidable. In this subsection we show a similar result for COP+Rel+TV+RA meaning representations and CQs by adapting his proof (i.e., his reduction) to COP+Rel+TV+RA with *indeterminate pronouns*, viz., "anybody", "somebody", etc., of semantics

$$\lambda P^{e\to t}.\forall x^e P(x) \quad \text{and} \quad \lambda P^{e\to t}.\exists x^e P(x).$$

Lemma 7.5.19. *There exists a set $\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}$ of COP+Rel+TV+RA sentences and a question Q expressing a CQ, s.t., for every tiling grid $\mathcal{T}$ there exists a tiling t for $\mathcal{T}$ iff $\tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}) \not\models \tau(Q)$*

Proof. Let $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$ be a tiling grid of $k+1$ tiles $\mathbb{T} := \{c_1, \ldots, c_k\}$ and $\mathbb{T}$ and $\mathbb{V}$ (horizontal and vertical) relations over $\mathbb{T}$. We can encode $\mathcal{T}$ with a set $\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}$ of COP+Rel+TV+RA sentences as follows.

We start by defining the set $\mathcal{S}_\mathcal{T}$ that encodes the tiling horizontal and vertical constraints. The transitive verbs H and V encode the horizontal $\mathbb{H}$ and vertical $\mathbb{V}$ relations resp., whereas each noun C_i encodes a tile $c_i \in \mathbb{T}$, for $i \in [0, k]$,:

$$\text{Everything } H\text{s something.} \tag{7.4}$$

$$\text{Everything } V\text{s something.} \tag{7.5}$$

$$\text{For all } 0 \leq i < j \leq k\text{: Anything that is not a } C_i \text{ is a } C_j. \tag{7.6}$$

$$\text{For all } 0 \leq i < j \leq k\text{: No } C_i \text{ is a } C_j. \tag{7.7}$$

$$\begin{array}{rc} \text{For all } (c, c') \notin \mathbb{V}\text{:} & \text{Everybody who } H\text{s somebody is a } C. \\ & \text{Everybody who is } H\text{d by somebody is a } C'. \end{array} \tag{7.8}$$

$$\begin{array}{rc} \text{For all } (c, c') \notin \mathbb{H}\text{:} & \text{Everybody who } V\text{s somebody is a } C. \\ & \text{Everybody who is } V\text{d by somebody is a } C'. \end{array} \tag{7.9}$$

$$\begin{array}{c} \text{Everybody who } \bar{H}\text{s somebody does not } H \text{ him.} \\ \text{Everybody who does not } H \text{ somebody } \bar{H}'\text{s him.} \end{array} \tag{7.10}$$

Next, we define $\mathcal{F}_\mathcal{T}$, that encodes the tiling proper:

$$\mathcal{F}_\mathcal{T} := \left\{ \begin{array}{cc} c_0 \text{ is a } C_0. & c_0 H\text{s } c_1. \\ \vdots & \vdots \\ c_k \text{ is a } C_k. & c_{k-1} H\text{s } c_k. \end{array} \right\}$$

Consider now the following Y/N-question that asks whether there exist (at least) four tiles for which the grid is not closed (grid closure is a necessary condition for a tiling to exist), question

$$\begin{array}{rcl} Q & := & \text{Does somebody } H \text{ somebody who } V\text{s somebody and } V \text{ somebody} \\ & & \text{such that the latter } \bar{H}s \text{ the former?} \end{array}$$

Whose meaning representation $\tau(Q)$ is the CQ

$$\varphi := \exists x \exists y \exists z \exists w (H(x, y) \wedge V(y, w) \wedge V(x, z) \wedge \bar{H}(z, w)),$$

Consider now the $\mathbf{FO}$ sentence

$$\chi := \forall x, y, z, w (H(x, y) \wedge V(x, z) \wedge V(y, w) \Rightarrow H(z, w))$$

and notice that the sentences in (7.10) from $\mathcal{S}_\mathcal{T}$ express an (explicit) definition for $\bar{H}$, that is, that the meaning representation

$$\tau(\text{Everybody who does not } H \text{ somebody } \bar{H}'\text{s him.})$$

and the meaning representation

$$\tau(\text{Everybody who } \bar{H}\text{s somebody does not } H \text{ him.})$$

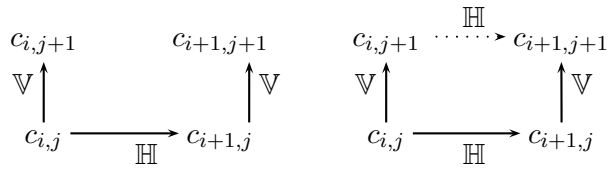


Figure 7.4: Sentence χ closes the grid, whereas query φ leaves it open.

together imply the **FO** axiom $\xi := \forall x, y(\bar{H}(x, y) \Leftrightarrow \neg H(x, y))$ and a fortiori imply that $\tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}) \not\models \varphi$ holds iff $\tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}) \cup \{\chi\}$ has a model.

We now claim that

$$\tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}) \cup \{\chi\} \text{ has a model} \quad \text{iff} \quad \text{there exists a tiling } t \text{ for } \mathcal{T}. \tag{†}$$

($\Rightarrow$) Let $\mathcal{I} = (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ be a model of $\tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T}) \cup \{\chi\}$. Define a mapping $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{D}_\mathcal{I}$ recursively as follows:

- For $i \in [0, k]$, $f(i, 0) := c_i^\mathcal{I}$.
- For $i \geq k$, $f(i + 1, 0) :=$ some c s.t. $(f(i, 0), c) \in H^\mathcal{I}$.
- For $i \geq k, j \geq 0$, $f(i + 1, j + 1) :=$ some c s.t. $(f(i, j), c) \in H^\mathcal{I}$.

Now, f is well-defined since *(i)* any grid point has always an $H^\mathcal{I}$-succesor (since $\mathcal{I} \models \tau((7.4)) \wedge \tau((7.5))$), *(ii)* the grid is always closed (since $\mathcal{I} \models \chi$) and *(iii)* $H^\mathcal{I}$ is non-empty (since $\mathcal{I} \models \mathcal{F}_\mathcal{T}$). Furthemore, by observing that $\mathcal{I}$ is modulo $\tau(\cdot)$ both a a model of (7.6)–(7.10) and a model of χ (and that hence $\mathcal{I} \not\models \varphi$), one can prove by double induction on $(i, j) \in \mathbb{N} \times \mathbb{N}$ that

- $(f(i, j), f(i, j + 1)) \in H^\mathcal{I}$ and
- $(f(i, j), f(i + 1, j)) \in V^\mathcal{I}$,

that is, f satisfies the horizontal and vertical constraints. Finally, to define the tiling $t\colon \mathbb{N} \times \mathbb{N} \to \mathbb{T}$ we put, for all $(i, j) \in \mathbb{N} \times \mathbb{N}$,

$$t(i, j) := c \text{ iff } f(i, j) \in C^\mathcal{I}.$$

($\Leftarrow$) For the converse let t be a tiling for $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$. We have to build a model $\mathcal{I}$ s.t. $\mathcal{I} \models \tau(\mathcal{S}_\mathcal{T} \cup \mathcal{F}_\mathcal{T})$ and $\mathcal{I} \not\models \varphi$. Define $\mathcal{I}$ as follows:

- $\mathbb{D}_\mathcal{I} := \mathbb{N} \times \mathbb{N}$.
- $H^\mathcal{I} := \{((i, j), (i, j + 1)) \in \mathbb{D}_\mathcal{I} \times \mathbb{D}_\mathcal{I} \mid ((i, j), (i, j + 1)) \in \mathbb{H}\}$.
- $V^\mathcal{I} := \{((i, j), (i + 1, j)) \in \mathbb{D}_\mathcal{I} \times \mathbb{D}_\mathcal{I} \mid ((i, j), (i + 1, j)) \in \mathbb{V}\}$.
- $C_i^\mathcal{I} := \{(i, 0) \in \mathbb{D}_\mathcal{I} \mid c_i \text{ is a } C_i \in \mathcal{F}_\mathcal{T}\}$, for $i \in [0, n]$.
- $c_i^\mathcal{I} := (i, 0)$, for $i \in [0, n]$.
- $\bar{H}^\mathcal{I} := (\mathbb{D}_\mathcal{I} \times \mathbb{D}_\mathcal{I}) \setminus H^\mathcal{I}$.

Clearly, $\mathcal{I}$ is the model we are looking for. □

Theorem 7.5.20. KBQA *is undecidable for COP+Rel+TV+RA when we consider arbitrary CQs.*

	Data		Combined	
	KBQA-UCQs	**KBSAT**	**KBQA-(U)CQs**	**KBSAT**
IS-$A^{\forall}_{\exists}$	in **LSpace** [Th 7.4.1]	in **LSpace** [Th 7.4.1]	**NPTime**-complete [Th 7.4.1]	in **NLSpace** [PHM09]
IS-$A^{\forall}_{\exists}$+TV	in **PTime** [Th 7.4.5]	in **LSpace** [Th 4.3.2]	**NPTime**-complete [Th 7.4.7]	**NLSpace**-complete [PHM09]
IS-$A^{\forall}_{\exists}$+DTV	N.A.	in **LSpace** [Th 7.5.6]	**NPTime**-hard [AHV95]	in **PTime** [PHT06]
IS-$A^{\forall}_{\exists}$+TV+DTV	N.A.	in **LSpace** [Th 7.5.6]	**NPTime**-hard [AHV95]	in **PTime** [PHT06]

	Data	
	KBQA-UCQs	**KBSAT**
COP	in **LSpace** [Th 4.3.2]	in **LSpace** [Th 7.5.6]
COP+TV	in **PTime** [Th 7.5.4]	in **LSpace** [Th 7.5.6]
COP+TV+DTV	N.A.	in **LSpace** [Th 7.5.6]

	Data	
	KBQA-(G)TCQs	**KBSAT**
COP+Rel	**coNPTime**-complete [PH08b]	in **LSpace** [Th 7.5.8]
COP+Rel+TV	**coNPTime**-complete [PH08b]	**NPTime**-complete [Th 7.5.13]
COP+Rel+DTV	**coNPTime**-complete [Th 7.5.12]	**NPTime**-complete [Th 7.5.13]
COP+Rel+DTV+TV	**coNPTime**-complete [Th 7.5.12]	**NPTime**-complete [Th 7.5.13]

Table 7.3: KBQA and . . . for the fragments of English and the positive fragments.

	Combined	
	KBQA-(U)CQs	**KBSAT**
COP	**NPTime**-complete [Th 4.3.6]	in **NLSpace** [PHM09]
COP+TV	**NPTime**-complete [Th 7.5.5]	**NLSpace**-complete [PHM09]
COP+TV+DTV	**NPTime**-hard [AHV95]	in **PTime** [PHT06]

	Combined	
	KBQA-(G)TCQs	**KBSAT**
COP+Rel	**coNPTime**-hard [PHT06]	**NPTime**-complete [PHT06]
COP+Rel+TV	**ExpTime**-complete [PHT06]	**ExpTime**-complete [PHT06]
COP+Rel+DTV	**coNExpTime**-hard [PHT06]	**NExpTime**-complete [PHT06]
COP+Rel+DTV+TV	**coNExpTime**-hard [PHT06]	**NExpTime**-complete [PHT06]

	KBQA-Atomic query	**SAT**
COP+Rel+TV+GA	Undecidable [Prop 7.5.17]	Undecidable [PHT06]
COP+Rel+DTV+TV+RA	Undecidable [Prop 7.5.17]	Undecidable [PHT06]
COP+Rel+DTV+TV+GA	Undecidable [Prop 7.5.17]	Undecidable [PHT06]

	KBQA-CQs	**SAT**
COP+Rel+TV+RA	Undecidable [Th 7.5.20]	**ExpTime**-complete [PHT06]

Table 7.4: KBQA and KBSAT for the fragments of English and the positive fragments (cntd).

7.6 Summary

In this chapter we have studied the data complexity of KBSAT and of KBQA in combination with (T)CQs for the fragments of English. Since the expressiveness of the fragments of English is orthogonal to that of known ontology languages, instead of using techniques coming from ontology languages, we chose instead the following strategy.

On the one hand, we studied the computational complexity of these problems for the so-called positive fragments (the fragments of English without negation), and showed that we can reduce KBQA for COP+TV to that of its positive counterpart, IS-$A^{\forall}_{\exists}$+TV and this one to **DATALOG**, for which data compelxity is known to be tractable.

Regarding **NPTime**-hard and/or **coNPTime**-hard (though decidable) fragments of English, we made use of resolution decision procedures in the spirit of Joyner that, we show, decide the $\exists^*\forall\exists^*$, $\exists^*\forall^2\exists^*$ and $\mathbf{SC}^+$ classes in time non-deterministic polynomial in the number of constants of their input knowledge bases or clause sets to derive **NPTime** and **coNPTime** data complexity upper bounds. This result can be applied to COP+TV+Rel, COP+DTV+Rel and COP+TV+DTV+Rel modulo a reduction (shown in [PHT06]) of their meaning representations to the monadic case that, we show, reduces the (un)satisfiability of their meaning representations to that of $\mathbf{SC}^+$ clauses.

We also strengthen the result by Pratt in [PH08b] on the undecidability of KBQA for arbitrary CQs over $\mathbf{FO}^2$ knowledge bases to COP+TV+DTV+Rel+RA knowledge bases (via a reduction from TP). Thus, anaphoric pronouns (even if restricted) both in the ontologies and in the queries/questions make query evaluation impossible to compute.

Tables 7.3 and 7.4 summarize the results of this chapter. Known results (and their corollaries) are stated together with a reference to the paper in which they were first published. New results refer to our theorems and their proofs.

Chapter 8

Conclusions

Computational, formal semantics provides a framework for studying both the algebraic and the combinatorial properties (resp. the semantic expressiveness and the semantic complexity) associated to the semantics of controlled languages. In particular, it allows to study the scalability of controlled languages and controlled language interfaces in ontology-based data access systems (OBDASs).

Scalability in OBDASs is influenced by the computational data complexity of the *(i)* query answering (KBQA) and *(ii)* knowledge base consistency (KBSAT) decision problems. By expressing, modulo a formal semantics compositional translation $\tau(\cdot)$, the query and ontology languages involved in KBQA and KBSAT in controlled language (i.e., by "reverse-engineering" controlled languages that map compositionally into exactly those query and ontology languages), controlled language scalability can be understood. Controlled languages possessing **PTime** or less data complexity scale to data. Controlled languages which possess a **coNPTime**- or **NPTime**-hard data complexity or higher cannot scale to data. Intractability arises when fragments become "Boolean closed", viz., capable of expressing complete sets of Boolean functions.

In Chapter 4 we expressed ontology and query languages which, like the *DL-Lite* family of description logics and graph-shaped conjunctive queries (GCQs) give rise to optimal data complexity (**LSpace**) for both KBQA and KBSAT. To this end, we defined the controlled languages Lite-English and GCQ-English. In particular, the distinction (critical for its good computational properties) between left and right concepts within the *DL-Lite* family is captured by syntactically subcategorizing constituents. Thus defined, they inherit the scalability of the the *DL-Lite* family and GCQs.

We studied also the relative and the absolute semantic expressiveness of Lite-English. In particular, we have shown that it overlaps in expressiveness with I. Pratt and A. Third's fragments of English, but that its absolute expressive power cannot be characterized. In Lite-English function words like "who" (i.e., relatives, expressing Boolean conjunction) can only occur in subject noun phrases ($\mathbf{NP}$s), while "not" (i.e., negations) can only occur in predicate verb phrases ($\mathbf{VP}$s). Furthermore, "every" and "no" (i.e., universal quantification) can only occur once and only in subject $\mathbf{NP}$s. Clearly, Lite-English is not "Boolean closed".

In Chapter 5 we extended GCQ-English to cover aggregations. More in general, we studied the issue of expressing SQL aggregate functions in controlled languages for OBDASs. Aggregate SQL queries (the SQL `SELECT-PROJECT-JOIN` fragment with `GROUP BY` and `HAVING` clauses, and aggregation functions such as `MAX`, `MIN`, `COUNT` or `SUM`) express a significant number of information requests to databases containing numerical data. In addition, corpus analysis showed that questions containing definite $\mathbf{NP}$s like, e.g., "the number of students", "the smallest integer", "the total number of men" (which combine with common nouns or nominals), which express intuitively such aggregation functions, can occur frequently. However, neither the semantics of SQL aggregation functions in OBDASs (or knowledge bases), nor the formal semantics of such

	Declarations	**Questions**
Constructs that scale (**PTime** or less)	– Negation in predicate **VP**s, relatives in predicate **VP**s, conjunction in predicate **VP**s. – Relatives and conjunction in subject **NP**s and predicate **VP**s, but no negation.	– Existential quantifiers, conjunction, relatives, aggregations, disjunctions.
Constructs that do not (**coNPTime**-hard)	– Negation in subject **NP**s. – Relatives and negation in subject **NP**s and predicate **VP**s predicate **VP**s	– Full negation. – Comparisons. – Universal restrictions.
Undecidable Constructs	– Transitive verbs, relatives, negation, existential and universal quantifiers, restricted anaphoric pronouns and indeterminate pronouns in subject **NP**s and predicate **VP**s, plus copula.	– Transitive verbs, existential indeterminate pronouns, relatives and restricted anaphoric pronouns.

Table 8.1: Combinations of controlled language constructs that scale to and do not scale to data w.r.t. OBDAS query evaluation. Note that the question constructs occur freely in **NP** and **VP** constituents.

definite English **NP**s were clear.

To tackle these two problems, we adopted the following strategy. On the one hand, we proposed a **FO**-based subset of SQL with aggregations, aggregate tree- and graph-shaped queries (ATCQs and AGCQs) and extended the certain answers semantics (and, accordingly, KBQA) of core SQL queries (viz., of SQL `SELECT-PROJECT-JOIN-UNION` queries) to cover these classes of queries. On the other hand, we provided a compositional semantic analysis of aggregations by proposing a class of *aggregate determiners* that express standard SQL aggregation functions. This gave way to the interrogative controlled language ATCQ-English that expresses ATCQs and AGCQs. Moreover, we showed that **HO** semantic analysis actually justifies our definition of certain answers. We also showed that aggregations are easy to compute. What does make query evaluation intractable (**coNPTime**-hard) are negations ("not"), universal quantification ("only") and comparatives ("higher than", "greater than", etc.) in the questions or formal queries.

In Chapter 6 we investigated the space of declarative controlled languages for which the data complexity of KBQA ranges from **LSpace** to **coNPTime**-hard. To this end, we introduced the IS-$A_{i\in[0,7]}$ family of controlled languages, orthogonal in expressiveness to Lite-English (and the *DL-Lite* family), EL-English (which expresses the description logic $\mathcal{ELI}$) and DL-English (which expresses the description logic $\mathcal{ALCI}$). The computational properties of the declarative controlled languages (and, accordingly, of the ontology languages) depend on whether we allow or not certain function words ("not", "only", "who", "some") to occur in either the subject **NP** or the predicate **VP** constituents of a sentence. Data complexity bounds were inferred by reasoning over the ontology languages expressed and/or induced by the resulting meaning representations. In particular, the IS-A_is induce a family of ontology languages for which the constructors of the concepts occurring to the left and to the right of the subsumption symbol $\sqsubseteq$ are non-symmetrical. This is achieved, once again, by subcategorizing controlled language constituents.

In Chapter 7 we studied the data complexity of KBSAT and of KBQA (with TCQs and/or GCQ-English questions as query language/interrogative controlled language) of I. Pratt's and A. Third's fragments of English, to see whether combinations of linguistically motivated declarative and interrogative fragments of English have better computational properties than description logic-based controlled languages. In these fragments, lexicon and syntax are restricted, but function words, with the exception of negation ("not") may occur within any constituent. Their expressive power thus depends on their coverage of English function words (and to a lesser degree, content words).

To study the data complexity of Pratt and Third's fragments we proposed several saturation-based resolution decision procedures. Such procedures allow to infer **coNPTime** and **NPTime** data complexity upper bounds for, resp., KBQA and KBSAT for all the fragments for which satisfiability is decidable. Our analysis showed, roughly, that tractability and scalability arise in the fragments containing "some", "every", with "no" and "not" restricted to (predicate) **VP**s, and containing no function words ("and", "who") or syntactic constructions (coordination, sentence subordination) expressing Boolean conjunction. When the latter are added, "Boolean closed" fragments are obtained and data complexity becomes intractable. In all cases, though, their data complexity is lower than their complexity for satisfiability. Going further, that is, allowing for restricted anaphoric pronouns, gives rise to undecidability.

More in general and as Table 8.1 summarizes, different combinations of constructs in the declarative controlled language (i.e., the ontology) and in the interrogative controlled language (i.e., the queries) give rise to different computational properties. Briefly, "Boolean closed" combinations, i.e., combinations expressing at the same time full Boolean negation (or complementation) and full Boolean conjunction (or intersection) are data intractable. When only restricted forms of conjunction and negation are expressed, the combinations remain tractable. Adding to a "Boolean

closed" combination restricted anaphors results in undecidability. Such construct-wise data complexity analysis can be used, we believe, as a basis for controlled language and controlled language interface design, insofar as intended for OBDASs, pinpointing the combination(s) of constructs to be covered when a compositional, efficient and completely accurate translation is targeted.

The table is to be read this way: each bullet states a certain combination of constructs in either the question or the declaration. In the top row we state maximal combinations of declarative and interrogative constructors (i.e., function words). In the middle row, the minimal intractable combinations. In the last row we state a minimal undecidable combination (strengthening Pratt's result for $\mathbf{FO}^2$ and SQL `SELECT-PROJECT-JOIN-UNION` queries).

To finish, we single out three possible future lines of research stemming from the results of this thesis. *(i)* To see whether empirical evaluations of the different intractable controlled language constructs do indeed blow-up an OBDAS's data access and management routines and if so, how frequently. *(i)* Consider controlled language questions with arbitrary generalized determiners like, e.g., "most", "a little", insofar as common in question corpora. *(iii)* Generalize the data complexity results from OBDASs to arbitrary incomplete databases.

Bibliography

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Welsey, 1995.

[ART95] Ioannis Androutsopoulos, Guy D. Ritchie, and Paul Thanisch. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.

[AUS86] Alfred V. Aho, Jeffrey D. Ullman, and Ravi Sethi. *Compilers. Principles, Techniques and Tools*. Addison Wesley - Longman, 1986.

[BB05a] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI, 2005.

[BB05b] Patrick Blackburn and Johan Bos. Working with discourse representation theory. An advanced course in computational semantics (draft). Available at `http://homepages.inf.ed.ac.uk/jbos/comsem/`, 2005.

[BBC+07] Raffaella Bernardi, Francesca Bonin, Domenico Carbotta, Diego Calvanese, and Camilo Thorne. English querying over ontologies: E-QuOnto. In *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence (AI*IA 2007)*, 2007.

[BBSS09] Dave Braines, Jie Bao, Paul R. Smart, and Nigel R. Shadbolt. A controlled natural language interface for semantic media wiki using the Rabbit language. In *Proceedings of the 2009 Controlled Natural Language Workshop (CNL 2009)*, 2009.

[BC80] John Barwise and Robin Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219, 1980.

[BCD05] Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.

[BCN+03] Franz Baader, Diego Calvanese, Daniele Nardi, Peter Patel-Schneider, and Deborah McGuinness. *The Description Logic Handbook*. Cambridge University Press, 2003.

[BCT07a] Raffaella Bernardi, Diego Calvanese, and Camilo Thorne. Expressing *DL-Lite* ontologies with controlled English. In *Proceedings of the 20th International Workshop on Description Logics (DL 2007)*, 2007.

[BCT07b] Raffaella Bernardi, Diego Calvanese, and Camilo Thorne. Lite natural language. In *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, 2007.

[Ber98] Arendse Bernth. Easy English: Addressing structural ambiguity. In D. Frawell et. al., editor, *Proceedings of the 1998 Machine Translation and the Information Soup Conference (AMTA 1998)*, 1998.

[BG01] Leo Bachmair and Harald Ganziger. *Resolution Theorem Proving*, volume 1 of *Handbook of Automated Reasoning*, chapter 2, pages 19–100. Elsevier - The MIT Press, 2001.

[BKGK05] Abraham Bernstein, Esther Kaufman, Anne Göring, and Christoph Kiefer. Querying ontologies: A controlled English interface for end-users. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, 2005.

[Blu99] Adam Blum. Microsoft English Query 7.5. automatic extraction of semantics from relational databases and OLAP cubes. In *Proceedings of the 25th International Conference on Very Large Databases (VLDB 1999)*, 1999.

[CAKZ07] Diego Calvanese, Alessandro Artale, Roman Kontchakov, and Michael Zakharyaschev. in the light of first order logic*DL-Lite*. In *AAAI-07*, 2007.

[Car97] Bob Carpenter. *Type-Logical Semantics*. The MIT Press, 1997.

[CdL+05a] Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. : Tractable description logics for ontologies*DL-Lite*. In *Proceedings of the 20th National Conference on Artificial Intelligence (AA*AI 2005)*, 2005.

[CdL+05b] Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Efficiently managing data intensive ontologies. In *Proceedings of the 2nd Italian Semantic Web Workshop: Semantic Web Applications and Perspectives (SWAP 2005)*, 2005.

[CdL+06] Diego Calvanese, Giuseppe de Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[CdV+06] Diego Calvanese, Giuseppe de Giacomo, Guido Vetere, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies: The description logic *DL-Lite_A*. In *Proceedings of the 2nd International Workshop on OWL: Experiences and Directions (OWLED-2006)*, 2006.

[CGL+07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[Che76] Peter Chen. The entity-relationship model - toward a unified view of data. *Journal of the ACM*, 1(1):355–389, 1976.

[CHH+08] Philipp Cimiano, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer. Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. *Data and Knowledge Engineering*, 65(2):325–354, 2008.

[CK90] Chen Chung Chang and Howard Jerome Keisler. *Model Theory*. Elsevier, 1990.

[CL03] René Cori and Daniel Lascar. *Logique mathématique (2 vols)*. Dunod, 2003.

[Cli88] James Clifford. Natural language querying of historical databases. *Computational Linguistics*, 14(1):10–35, 1988.

[CLN99] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, 1977.

[CNKT08] Diego Calvanese, Werner Nutt, Evgeny Kharlamov, and Camilo Thorne. Aggregate queries over ontologies. In *Proceedings of the 2nd International Workshop on Ontologies and Information Systems for the Semantic Web (ONISW 2008)*, 2008.

[CNS03] Sara Cohen, Werner Nutt, and Yeshoshua Sagiv. Containment of aggregate queries. In *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*, 2003.

[CNS07] Sara Cohen, Wernet Nutt, and Yeshoshua Sagiv. Deciding equivalences among conjunctive aggregate queries. *Journal of the ACM*, 54(2):1–50, 2007.

[CV93] Surajit Chaudhuri and Moshe Vardi. Optimization of real conjunctive queries. In *Proceedings of the 12th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS 1993)*, 1993.

[DMB03] Micheal Dittenbach, Dieter Merkl, and Helmut Berger. Querying tourism information systems in natural language. In *Proceedings of the 2nd International Conference on Information Systems Technology and its Applications (ISTA 2003)*, 2003.

[EGDV01] Thomas Eiter, Georg Gottlob, Evgeny Dantsin, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[EN04] Ramez Elmasri and Shamkant B. Navanthe. *Fundamentals of Database Systems*. Addison Welsey, 2004.

[FK06] Norbert E. Fuchs and Kaarel Kaljurand. Mapping Attempto Controlled English to OWL-DL. In *Demos and Posters of the 3rd European Semantic Web Conference (ESWC 2006)*, 2006.

[FKS05] Norbert E. Fuchs, Kaarel Kaljurand, and Gerold Schneider. Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2006)*, 2005. Available at `http://www.ifi.unizh.ch/attempto/publications/`.

[FLHT01] Christian G. Fermüller, Alexander Leitsch, Ullrich Hustadt, and Tanel Tammet. *Resolution Decision Procedures*, volume 2 of *Handbook of Automated Reasoning*, chapter 2, pages 1791–1849. Elsevier - The MIT Press, 2001.

[FS95] Norbert Fuchs and Rolf Schwitter. Specifying logic programs in controlled natural language. In *Proceedings of the 1995 Workshop on Computational Logic for Natural Language Processing (CLNLP 1995)*, 1995.

[FSH+05] Norbert Fuchs, Uta Schwertel, Stefan Hoeffer, Kaarel Kaljurand, and Gerold Schneider. Extended discourse representation structures in Attempto Controlled English. Technical report, University of Zurich, 2005. Available at `http://www.ifi.unizh.ch/attempto/publications/`.

[Gam91] L. T. F. Gamut. *Logic, Language and Meaning (2 vols.)*. University of Chicago Press, 1991.

[GGB01] Yuri Gurevitch, Erich Grädel, and Egon Börger. *The Classical Decision Problem*. Springer, 2001.

[GHLS07] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007.

[Gio08] Alessandra Giordani. Mapping natural language into SQL in a NLIDB. In *Proceedings of the 13th International Conference on Applications of Natural Language to Information Systems (NLDB 2008)*, 2008.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[GKV97] Erich Grädel, Phokion Kolaitis, and Moshe Vardi. On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, (1), 1997.

[GM09] Alessandra Giordani and Alessandro Moschitti. Syntactic structural kernels for natural language interfaces to databases. In *Proceedigs of the 2nd European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2009)*, 2009.

[Grä99] Erich Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[Gua98] Nicola Guarino. Formal ontology and information systems. In *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS 1998)*, 1998.

[HK98] Irene Heim and Angelika Kratzer. *Semantics in Genarative Grammar*. Blackwell, 1998.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[HLNW99] Lauri Hella, Leonid Libkin, Juha Nurmonen, and Limsoon Wong. Logics with aggregate operators. In *Proceedings of the 14th Symposium on Logic in Computer Science (LICS 1999)*, 1999.

[HPSv03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal on Web Semantics*, 1(1):7–26, 2003.

[HSG04] Ullrich Hustadt, Renate Schimdt, and Lilia Georgieva. A survey of decidable first-order fragments and description logics. *Journal on Relational Methods in Computer Science*, 1(1), 2004.

[HT02] Ian Horrocks and Sergio Tessaris. Querying the semantic web: A formal approach. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2002)*, 2002.

[JM09] Daniel Jurafsky and James Martin. *Speech and Language Processing*. Prentice Hall, 2nd edition, 2009.

[Jr.76] William H. Joyner Jr. Resolution strategies as decision procedures. *Journal of the ACM*, 23(3):398–417, 1976.

[Kal07] Kaarel Kaljurand. *Attempto Controlled English as a Semantic Web Language*. PhD thesis, University of Tartu, 2007. Available at `http://attempto.ifi.uzh.ch/site/pubs/`.

[Kar77] Lauri Karttunen. Syntax and semantics of questions. *Linguistics and Philosophy*, 1(1):3–44, 1977.

[KB07] Esther Kaufmann and Abraham Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *Proceedings of the 6th International Web Conference and the 2nd Asian Web Conference (ISWC/ASWC 2007)*, pages 281–294, 2007.

[KF06] Kaarel Kaljurand and Norbert E. Fuchs. Birectional mapping between OWL-DL and attempto controlled english. In *Proceedings of the 4th International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2006)*, 2006.

[KF07] Kaarel Kaljurand and Norbert E. Fuchs. Verbalizing OWL in attempto controlled english. In *Proceedings of 3rd OWL Experiences and Directions Workshop (OWLED 2007)*, 2007.

[KRv05] Hans Kamp, Uwe Reyle, and Josef van Genabith. Discourse representation theory. To appear in the new edition of the *Handbook of Philosophical Logic*. Available at `http://www.ims.uni-stuttgart.de/~hans/`, 2005.

[Lal97] René Lalément. *Logique, réduction, résolution*. Dunod, 1997.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.

[Len02] Maurizio Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, New York, NY, USA, 2002. ACM.

[LK07] Carsten Lutz and Adila Krisnadhi. Data complexity in the $\mathcal{EL}$ family of DLs. In *Proceedings of the 20th International Workshop on Description Logics (DL 2007)*, 2007.

[MC99] Frank Meng and Wesley Chu. Database query formation from natural language using semantic modeling and statistical keyword meaning disambiguation. Technical report, University of California at Los Angeles, 1999. Available at `http://www.cobase.cs.ucla.edu/publications.html`.

[MHWB06] Sela Mador-Haim, Yoad Winter, and Anthony Braun. Controlled language for geographical information system queries. In *Proceedings of Inference in Computational Semantics 2006*, 2006.

[Min05] Michael Minock. A Phrasal Approach to Natural Language Interfaces over Databases. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, 2005.

[Mon70] Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[Mon73] Richard Montague. The proper treatment of quantification in ordinary english. In *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, 1973.

[MON08] Michael Minock, Peter Olofson, and Alexander Näslund. Towards building robust natural language interfaces to databases. In *Proceedings of 13th International Conference on Applications of Natural Language to Information Systems (NLDB 2008)*, 2008.

[Moo97] Michael Moortgat. *Categorial Type Logics*, chapter 2, pages 93–177. Handbook of Logic and Language. Elsevier, 1997.

[OCE08] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data complexity of query answering in expressive description logics. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[Pap94] Christos Papadimitrou. *Computational Complexity*. Addison Wesley - Longman, 1994.

[PEK03] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the Conference on Intelligent User Interfaces, 2003.*, 2003.

[PH01] Ian Pratt-Hartmann. On the semantic complexity of some fragments of English. Technical report, University of Manchester, 2001. Available at `http://www.cs.manchester.ac.uk/~ipratt/`.

[PH04] Ian Pratt-Hartmann. Fragments of language. *Journal of Logic, Language and Information*, 13(2):207–223, 2004.

[PH08a] Ian Pratt-Hartmann. Computational complexity in natural language. To appear in the *Handbook of Computational Linguistics and Natural Language Processing*. Available at `http://www.cs.manchester.ac.uk/~ipratt/`, 2008.

[PH08b] Ian Pratt-Hartmann. Data complexity of the two-variable fragment with counting quantifiers. *Information and Computation*, 207(8):867–888, 2008.

[PH09] Ian Pratt-Hartmann. Computational complexity of controlled natural languages. In *Pre-Proceedings of the Workshop in Controlled Languages (CNL 2009)*, 2009.

[PHM09] Ian Pratt-Hartmann and Larry Moss. Logics for the relational syllogistic. *Review of Symbolic Logic*, 4(2):647–683, 2009.

[PHT06] Ian Pratt-Hartmann and Allan Third. More fragments of language. *Notre Dame Journal of Formal Logic*, 47(2):151–177, 2006.

[PWt93] Barbara H. Partee, Robert Wall, and Alice ter Meulen. *Mathematical Methods in Linguistics*. Kluwer, 1993.

[Rob65] Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1), 1965.

[Ros07] Riccardo Rosati. The limits of querying ontologies. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, 2007.

[Sch93] Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278, 1993.

[Sch05] Uta Schwertel. *Plural Semantics for Natural Language Understanding*. PhD thesis, Faculty of Arts – University of Zurich, 2005. Available at `http://www.ifi.unizh.ch/attempto/publications`.

[Sch08] Rolf Schwitter. Creating and querying linguistically motivated ontologies. In *Proceedings of the 2008 Knowledge Representation Workshop (KROW 2008)*, 2008.

[SKC+08] Rolf Schwitter, Kaarel Kaljurand, Anne Cregan, Catherine Dolbear, and Glen Hart. A comparison of three controlled natural languages for owl 1.1. In *Proceedings of the 4th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, Washington, 1–2 April 2008.

[Sla07] Marija Slavkovik. Deep analysis for an interactive question answering system. Master's thesis, Free University of Bozen - Bolzano, 2007.

[SLH03] Rolf Schwitter, Anna Ljungberg, and David Hood. ECOLE - a look-ahead editor for a controlled language. In *Proceedings of the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Applications Workshop (EAMT/CLAW 2003)*, 2003.

[Sow99] John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks Cole Publishing Co., 1999.

[Sow04] John Sowa. Common Logic Controlled English (draft). Available at `http://www.jfsowa.com/clce/specs.htm`, 2004.

[SS04] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.

[ST06] Rolf Schwitter and Marc Tilbrook. Let's talk in description logic via controlled language. In *Proceedings of the 3rd International Workshop on Logic and Engineering of Natural Language Semantics (LENLS 2006)*, 2006.

[Str05] Lutz Straßburger. What is a logic, and what is a proof? In Jean-Yves Beziau, editor, *Logica Universalis*, pages 135–145. Birkhauser, 2005.

[TBC+07] Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, Brian Davis, Siegfried Handschuh, and Adam Funk. CLOnE: Controlled language for ontology editing. In *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC 2007)*, 2007.

[TC09] Camilo Thorne and Diego Calvanese. Tree-shaped aggregate queries over ontologies. In *Proceedings of the 8th International Conference on Flexible Query Answering Systems (FQAS 2009)*, 2009.

[TC10a] Camilo Thorne and Diego Calvanese. Controlled English ontology-based data access. In *Proceedings of the 2009 Workshop on Controlled Natural Language (CNL 2009)*, 2010.

[TC10b] Camilo Thorne and Diego Calvanese. The data complexity of the syllogistic fragments of English. In *Proceedings of the 2009 Amsterdam Colloquium (AC 2009)*, 2010.

[Thi06] Allan Third. *The Logical Analysis of Fragments of Natural Language*. PhD thesis, School of Computer Science – University of Manchester, 2006. Available at `www.cs.man.ac.uk/~ipratt/theses`.

[Tho08] Camilo Thorne. Expressing aggregate queries over *DL-Lite* ontologies with controlled English. In *Proceedings of the 2008 European Summer School in Logic Language and Computation Student Session (ESSLLI 2008)*, 2008.

[TS00] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge U. Press, 2000.

[UHM01] Jeffrey D. Ullman, John E. Hopcroft, and Rajeev Motwani. *Introduction to Automata Theory, Languages and Computation*. Addison Welsley, 2001.

[Unw05] Mike Unwalla. AECMA Simplified English (draft). Available at `http://www.simplifiedenglish-aecma.org/`, 2005.

[Var82] Moshe Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, 1982.

Index

Curriculum Vitae and Publications

Camilo Thorne was born in January 8, 1975 in Lima (Peru). He finished his BA in Philosophy in 1997 at the PUCP (Pontificia Universidad Católica del Perú) University, Lima, Peru, where he graduated first of his class. In 2003 he finished a French Maitrise in Logic at Paris 1 University, Paris, France. In 2004 he finished a DEA (Diplôme d'Etudes Approfondies) in Artificial Intelligence and Optimization at Paris 13 University, Villetaneuse, France, during which he spent 6 months as an intern at the CNRS (Centre National de la Recherche Scientifique) LIPN (Laboratoire d'Informatique de Paris Nord) Laboratory in Villetaneuse. After several internships in Trento, Italy, at the LOA (Laboratory of Applied Ontology), and in Milton Keynes, UK, at the KMi (Knowledge Media Institute), he enrolled in 2005 in the PhD program in Computer Science of the Free University of Bozen-Bolzano, where he is currently research fellow (assegnista di ricerca).

List of Publications

Peer-reviewed

- Camilo Thorne, Diego Calvanese – "Controlled English ontology-based data access" – 2010, *Proceedings of the CNL 2009 Workshop*.
- Camilo Thorne, Diego Calvanese – "The data complexity of the syllogistic fragments of English" – 2010, *Proceedings of the AC 2009 Colloquium*.
- Camilo Thorne, Diego Calvanese – "Tree-shaped aggregate questions over ontologies" – 2009, *Proceedings of the FQAS 2009 Conference*.
- Camilo Thorne, Diego Calvanese – "Exploring controlled English ontology-based data access" (extended abstract) – 2009, *Pre-Proceedings of the CNL 2009 Workshop*.
- Camilo Thorne – "Expressing conjunctive and aggregate queries over ontologies with controlled English" – 2008, *Proceedings of the ESSLLI 2008 Student Workshop*.
- Diego Calvanese, Evgeny Kharlamov, Werner Nutt, Camilo Thorne – "Aggregate queries over ontologies" – 2008, *Proceedings of the ONISW 2008 Workshop*.
- Raffaella Bernardi, Francesca Bonin, Diego Calvanese, Domenico Carbotta, Camilo Thorne – "English querying over ontologies: E-QuOnto" – 2007, *Proceedings of the AI*IA 2007 Conference*.
- Camilo Thorne – "Managing structured data with controlled English - An Approach based on Description Logics" – 2007, *Proceedings of the ESSLLI 2007 Student Workshop*.
- Raffaella Bernardi, Diego Calvanese, Camilo Thorne – "Expressing *DL-Lite* ontologies with controlled English" – 2007, *Proceedings of the DL 2007 Workshop*.

- Raffaella Bernardi, Diego Calvanese, Camilo Thorne – "Lite natural language" – *Proceedings of the IWCS 2007 Workshop*.

Technical reports

- Camilo Thorne – *Equivalence of Aggregate Queries with Incomplete Information and Dependencies* - 2008, KRDB report.
- Raffaella Bernardi, Diego Calvanese and Camilo Thorne – *Querying Structured Data with Lite Natural Language* – 2008, KRDB report.
- Camilo Thorne – *Expressive Power of DL-Lite (II)* – 2008, KRDB report.
- Camilo Thorne – *Controlled English for DL-Lite* – 2007, KRDB report.

Other

- Camilo Thorne – "Categorial module grammars have finite bounded density" – 2007, *Proceedings of LATA 2007*.
- Camilo Thorne, Victoria Uren, Jianhan Zhu – *Extracting Domain Ontologies with CORDER* - 2005, KMi report.
- Camilo Thorne – "Sobre el imperativo categórico como regla de universalización" – 2000, *Proceedings of the 7th Peruvian Congress of Philosophy*.

Titles in the KRDB Dissertation Series

KRDB DS-2008-01: **Catharina Maria Keet**
A Formal Theory of Granularity

KRDB DS-2008-02: **Dino Seppi**
Prosody in Automatic Speech Processing

KRDB DS-2010-01: **Manuel Kirschner**
The Structure of Real User-System Dialogues in Interactive Question Answering

KRDB DS-2010-02: **Lina Lubyte**
Techniques and Tools for the Design of Ontologies for Data Access

KRDB DS-2010-03: **Ornella Mich**
Usability Methods and Deaf Children. The Case of the LODE e-tool

KRDB DS-2010-04: **Mariano Rodriguez-Muro**
Tools and Techniques for Ontology Based Data Access in Lightweight Description Logics

KRDB DS-2010-05: **Vladislav Ryzhikov**
Temporal Extensions of DL-Lite

KRDB DS-2010-06: **Camilo Thorne**
Query Answering over Ontologies Using Controlled Natural Languages

For further information about KRDB publications, please contact

KRDB Research Centre for Knowledge and Data
Piazza Domenicani 3
39100 Bozen-Bolzano
Italy

tel: +39 0471 016 120
fax: +39 0471 016 009
homepage: `http://www.unibz.it/inf/krdb`