Ontology-Mediated Querying with Horn Description Logics

Dissertation

Leif Sabellek, M. Sc.

Universität Bremen
Fachbereich 3 - Mathematik/Informatik

Betreuer

Prof. Dr. Carsten Lutz

Gutachter

Prof. Dr. Carsten Lutz
Prof. Dr. Frank Wolter

Datum der Verteidigung

28. August 2019

Abstract

An *ontology-mediated query* (OMQ) consists of a database query paired with an ontology. When evaluated on a database, an OMQ returns not only the answers that are already in the database, but also those answers that can be obtained via logical reasoning using rules from ontology.

There are many open questions regarding the complexities of problems related to OMQs. Motivated by the use of ontologies in practice, new reasoning problems which have never been considered in the context of ontologies become relevant, since they can improve the usability of ontology enriched systems. This thesis deals with various reasoning problems that occur when working with OMQs and it investigates the computational complexity of these problems. We focus on ontologies formulated in Horn description logics, which are a popular choice for ontologies in practice. Specifically, we cover the following three topics:

- We classify all OMQs based on a conjunctive query and an $\mathcal{EL}$ ontology regarding their data complexity and rewritability into linear Datalog. We show that evaluation of such OMQ is either in AC^0, NL-complete or PTime-complete. Additionally, we show that it is ExpTime-complete to decide in which of the three categories a given OMQ falls. Finally, we show that such an OMQ is rewritable into linear Datalog if and only if it can be evaluated in NL.
- We investigate the problem *query-by-example* where given a knowledge-base and positive and negative examples of query answers, a conjunctive query should be computed whose answers generalize the given examples. We analyze this problem for ontologies formulated in Horn-$\mathcal{ALC}$ and $\mathcal{ELI}$. While the problem is coNExpTime-complete for Horn-$\mathcal{ALC}$, it becomes undecidable for $\mathcal{ELI}$.
- In *ontology-based data access*, multiple databases are joined by so-called *mappings* into a global schema, which is then additionally enriched by an ontology. We investigate the question whether a given query over the original databases can also be answered via a query formulated over the global schema. We show that this problem is Π_2^p-complete for DL-Lite ontologies and coNExpTime-complete or 2-ExpTime-complete for $\mathcal{ELHI}$ ontologies, depending on whether the query is rooted or not.

Zusammenfassung

Eine *ontology-mediated query* (OMQ) ist eine Datenbankanfrage gepaart mit einer Ontologie. Bei der Auswertung einer OMQ auf einer Datenbank werden nicht nur die Antworten geliefert, die bereits in der Datenbank stehen, sondern auch solche, die mittels Regeln aus der Ontologie geschlussfolgert werden können.

Es gibt viele offene Fragen, was die Komplexität diverser Schlussfolgerungsprobleme im Zusammenhang mit OMQs anbelangt. Motiviert durch den Einsatz von Ontologien in der Praxis werden auch immer wieder neue Schlussfolgerungsprobleme relevant, deren Anwendung die Benutzerfreundlichkeit verbessern kann, die aber bisher nie im Zusammenhang mit Ontologien untersucht worden sind. Diese Arbeit beschäftigt sich mit der Analyse diverser Schlussfolgerungsprobleme, die im Zusammenhang mit OMQs auftreten und untersucht diese Probleme in Bezug auf ihre Berechnungskomplexität. Dabei fokussieren wir uns auf Ontologien, die in Horn-Beschreibungslogiken formuliert sind, welche eine beliebte Wahl für Ontologien in der Praxis darstellen. Konkret behandeln wir folgende drei Themen:

- Wir klassifizieren sämtliche OMQs basierend auf einer konjunktiven Anfrage und einer $\mathcal{EL}$-Ontologie bzgl. ihrer Datenkomplexität und Umformulierbarkeit in linear Datalog. Wir zeigen, dass jede solche OMQ entweder enthalten in AC^0, NL-vollständig oder PTime-vollständig ist. Zusätzlich zeigen wir, dass es ExpTime-vollständig ist zu entscheiden, in welche der drei Kategorien eine gegebene OMQ fällt. Außerdem zeigen wir, dass eine solche OMQ genau dann in linear Datalog umformuliert werden kann, wenn sie in NL ausgewertet werden kann.
- Wir untersuchen das Problem *query-by-example*, bei dem zu gegebenen positiven und negativen Beispielen eine konjunktive Anfrage berechnet werden soll, deren Antworten die gegebenen Beispiele verallgemeinern. Dieses Problem analysieren wir für Ontologien formuliert in Horn-$\mathcal{ALC}$ und $\mathcal{ELI}$. Während das Problem für Horn-$\mathcal{ALC}$ noch coNExpTime-vollständig ist, ist es für $\mathcal{ELI}$ unentscheidbar.
- Unter dem Stichwort *ontology-based data access* werden mehrere Datenbanken durch sogenannte *mappings* in einem globalen Schema zusammengeführt, welches dann zusätzlich durch eine Ontologie bereichert wird. Wir untersuchen die Frage, ob eine gegebene Anfrage über den ursprünglichen Datenbanken auch über dem globalen Schema beantwortet werden kann, wenn die Ontologie in $\mathcal{ELHI}$ oder der DL-Lite-Familie formuliert ist. Wir zeigen, dass dieses Problem für DL-Lite-Ontologien Π_2^p-vollständig ist und für $\mathcal{ELHI}$-Ontologien coNExpTime-vollständig oder 2-ExpTime-vollständig, abhängig davon, ob die Anfrage rooted ist oder nicht.

Acknowledgements

First of all, I want to thank my advisor Carsten Lutz. He introduced me to the area of description logics and did a truly fantastic job advising me over the past three and a half years. I feel very lucky to be part of his group *Theorie der künstlichen Intelligenz* which provided me with a pleasant working environment. I am very grateful to my colleagues, TDKIers and collaborators Cristina Feier, Víctor Gutiérrez Basulto, Anneke Haga, Peter Hansen, Jean Christoph Jung, Antti Kuusisto, Mauricio Martel, Johannes Marti and Thomas Schneider for countless interesting discussions. It is always a pleasure working with you and I learned a lot from you! Furthermore, I thank Frank Wolter for agreeing to review my thesis.

I also thank my brother André Sabellek for giving me feedback on the introduction chapter. For the beautiful style this document is set in, my warmest thanks goes to Christoph Pegel, who helped me setting up the document in the same style that he used for his own thesis. I owe thanks to my parents, who made it possible for me to study without any worries, and finally, I thank Emily for her everlasting and loving support.

Bremen, July 2019
Leif Sabellek

Contents

1 Introduction

Relational databases are the classical formalism to digitally represent data. A relational database consists of relations (also called tables), where a relation is a set of facts of the same arity. Traditionally, databases are interpreted under the *closed world assumption*, which means that a fact is considered to be true if and only if it is contained in the database [Rei77]. In other words: Everything that is not explicitly written in the database is considered to be false. But this approach has some limitations: First, in many scenarios, especially in times of Big Data, one has to manage incredible amounts of data that arise from multiple sources, scattered across many different databases. This means one has to deal with incomplete and inhomogeneous knowledge. Thus, we cannot assume that all true facts are in our database, but merely consider the content of the database as a collection of facts that we know so far. Second, there might be a lot of knowledge that is not explicitly stored in the database to reduce redundancy, but that could be derived from the database using logical reasoning by someone familiar with the matter. For these purposes, a different approach to the usage of databases has emerged, which is called the *open world assumption*. Under the open world assumption, one interprets the facts in the database as true, but there might be more true facts that can be derived via logical reasoning using background knowledge. This background knowledge is stored in a so-called *ontology*.

An ontology is a set of logical sentences which represent general knowledge about a specific domain. Ontologies are very popular in the fields of biology and medicine, since these fields are home to large amounts of pure factual knowledge. While a database also stores knowledge, the main difference is that a database stores *instance data*, which is data about specific situations that could also change over time. For example, a database in a hospital could store information about patients, doctors, rooms, appointments, and diagnoses. An ontology on the other hand stores general knowledge, like knowledge about anatomy, diseases or medicine. Together, the database and the ontology form a so-called *knowledge base*.

A knowledge base contains a lot of implicit knowledge that can be derived from the database in combination with the ontology. If queries are posed in the presence of an ontology, one usually considers the query and the ontology together as a compound query, a so-called *ontology-mediated query (OMQ)*. When answering an OMQ, one does not simply speak of *answers* to the query, but of *certain answers*, which are all answers to the query that can be derived from the database using the ontology and logical reasoning. This approach has been studied extensively, see for example [Cal+13; CGP12; Bie+14]. Consider the following example:

Example 1.1. *An ontology about diseases, formulated in the description logic $\mathcal{EL}$, could* *include the following rules:*

$$\begin{aligned}\mathsf{AlzheimerDisease} &\sqsubseteq \mathsf{DementiaDisorder}\\ \mathsf{DementiaDisorder} &\sqsubseteq \exists\,\mathsf{hasFindingSite}.\mathsf{BrainPart}\\ \mathsf{BrainConcussion} &\sqsubseteq \exists\,\mathsf{hasFindingSite}.\mathsf{BrainPart}\end{aligned}$$

The first rule says that the Alzheimer's disease is a dementia disorder. The second rule says that every instance of DementiaDisorder *is related to an instance of* BrainPart *via the binary relation* hasFindingSite. *The third rule states the same about* BrainConcussion. *Assume a hospital's database includes the following facts:*

$$\begin{aligned}&\mathsf{hasFinding}(\mathsf{patient12}, \mathsf{finding345})\\ &\mathsf{hasFinding}(\mathsf{patient45}, \mathsf{finding257})\\ &\mathsf{AlzheimerDisease}(\mathsf{finding345})\\ &\mathsf{BrainConcussion}(\mathsf{finding257})\end{aligned}$$

A doctor needs a list of all patients who have a finding located in the brain. Then the OMQ consisting of the ontology above and the query

$$q(x) \leftarrow \mathsf{hasFinding}(x, y) \land \mathsf{hasFindingSite}(y, z) \land \mathsf{BrainPart}(z)$$

returns both patient12 *and* patient45 *as certain answers.*

Example 1.1 also exemplifies another advantage of ontologies which is introducing new vocabulary that can be used for querying. This is especially useful in the context of *ontology-based data access (OBDA)*, where data from multiple databases with different schemes are unified using an ontology, which provides a new, global schema and relates the global schema to all the source schemas [Pog+08].

Description Logics

How can ontologies be formally defined? First-order logic (FO) is the most fundamental logic in computer science and it seems natural to define an ontology to be just any set of sentences in first-order logic. However, it is not viable to use full FO for formulating ontologies because central reasoning problems like satisfiability and consequence are undecidable for FO. So one has to find weaker fragments of FO whose reasoning problems are decidable but such that they still provide enough expressive power to formulate meaningful ontologies. Searching and investigating such fragments has been a topic of research over the last thirty years and different families of fragments have been identified, which are referred to as *description logics (DLs)*.

Description logics (DLs) are decidable fragments of first order logic that have become a popular choice for formulating ontologies [Baa+07; Baa+17]. It is notable that DLs only use unary and binary predicates, where unary predicates are called *concept names* and binary predicates are called *roles*.[1] In the Example 1.1, AlzheimerDisease, DementiaDisorder, BrainConcussion and BrainPart are concept names, and hasFinding and hasFindingSite are roles. Depending on the specific DL, different sets of operators can be used to form *concepts*, which correspond to FO formulas with one free variable. In the example, $\exists$ hasFindingSite.BrainPart is a concept that describes all objects which are related via the role hasFindingSite to an instance of the class BrainPart.

There is a large variety of of DLs with different expressive power and complexity of reasoning. Very expressive DLs like for instance $\mathcal{SHOIQ}$ can express, among others,

- disjunctions of concepts ('every human is dead or alive'),
- transitivity of roles ('if x is a part of y and y is a part of z, then x is a part of z),
- role hierarchies ('if x is the father of y, then x is a parent of y'),
- inverse roles ('if x is the father of y, then y is a child of x'),
- number restrictions ('every hand has five fingers') and
- can refer to concrete individuals ('everyone knows Dave').

More inexpressive DLs like $\mathcal{EL}$ on the other hand only allow simple rules like

- concept name inclusion ('every student is a person'),
- conjunction (if x is a person and x is female, then x is a woman') and
- existential restrictions ('if x has a mother that is a chimpanzee, then x is a chimpanzee' or 'every country has a capital city').

The reason to consider a large variety of DLs is the trade-off between expressive power and computational complexity. The more expressive the logic, the harder the reasoning problems become. To give a rough idea: Many standard reasoning problems for expressive DLs like $\mathcal{SHOIQ}$ are NExpTime-complete [Tob01] or of even higher complexity, while for less expressive DLs like $\mathcal{EL}$ or the DL-Lite family, many reasoning problems are solvable in PTime or coNP [Art+09; KL07]. But if the logic is not expressive enough, it might not be suitable to model the knowledge appropriately. It turns out that the complexity is crucially influenced by whether or not disjunctions are allowed. The explanation is simple: Disjunctions do not allow unique conclusions to be drawn. So while the other mentioned types of conclusion rules can essentially be applied in only one way, leading to a unique result, disjunctive rules lead to different results and it becomes harder to check whether a certain fact is logically implied by the rules. In fact, one can encode NP-complete problems into the OMQ answering problem even for a fixed ontology, if the ontology language allows disjunctions. This is shown in the following example, using an ontology formulated in the description logic $\mathcal{ALC}$:

[1]Reasoning with only unary and binary relations is useful for several reasons: The data can be visualized as labelled graphs and is thus easier to understand. Also, relations of higher arity are not always necessary. For example, instead of using a 4-ary relation Appointment(id, date, doctor, patient), one can use four concept names Appointment, Date, Doctor, Patient and three roles hasDate, hasDoctor, hasPatient to represent the same data.

Example 1.2. *[Baa+17] The 3-colorability problem is a well-known NP-complete problem. It asks whether the vertices of a given graph can be colored with three different colors in such a way that no two adjacent vertices have the same color. Let the graph be stored as a database with a single binary relation E for the edges of the graph and fix the following ontology:*

$$\begin{aligned} \top &\sqsubseteq \mathsf{Red} \sqcup \mathsf{Blue} \sqcup \mathsf{Green} \\ \mathsf{Red} \sqcap \exists\, E.\mathsf{Red} &\sqsubseteq \mathsf{Conflict} \\ \mathsf{Blue} \sqcap \exists\, E.\mathsf{Blue} &\sqsubseteq \mathsf{Conflict} \\ \mathsf{Green} \sqcap \exists\, E.\mathsf{Green} &\sqsubseteq \mathsf{Conflict} \end{aligned}$$

The first rule states that every vertex is colored either red, blue or green. The other three rules express that whenever a vertex has a neighbour of the same color, then this vertex is labelled with the concept name Conflict. *The query*

$$q() \leftarrow \mathsf{Conflict}(x)$$

asks whether there is a vertex labelled with Conflict, *so the query evaluates to true if and only if the graph is not 3-colorable.*

For this reason, DLs without disjunctions are investigated. These DLs are called *Horn DLs* and they are a popular choice as ontology languages. Widely used ontologies like SNOMED CT (Systematized Nomenclature of Human and Veterinary Medicine – Clinical Terms) and GALEN (Generalised architecture for languages, encyclopedia and nomenclatures in medicine) are to a great extent formulated in a Horn DL. The ontology language used in Example 1.1, $\mathcal{EL}$, is also a Horn DL. Horn DLs enjoy nice properties, most important for answering OMQs is the universal model property: It is possible to apply conclusion rules from the ontology in a straightforward way to obtain a (generally infinite) extension of the database (the so-called *universal model*) which includes all facts that are relevant for answering certain types of queries, so that OMQs can be answered by constructing the universal model and then evaluating the query as a standard (not ontology-mediated) query on the universal model.

Reasoning problems

There are a lot of open questions regarding OMQs with Horn DLs. This thesis contributes to foundational research about Horn DLs, more precisely we are concerned with pinpointing the computational complexity of several decision problems involving OMQs. We focus on two areas:

1. Get a deeper understanding of the complexities of answering Horn DL OMQs.
2. Introduce new relevant reasoning problems and analyse their complexities.

In the following, we give a broad overview of all reasoning problems that are studied in this thesis. A more detailed overview including more references to related work can be found at the beginning of the relevant chapter. For all the following reasoning problems, the ontology languages we consider are always one or several different Horn DLs.

Data complexity of answering OMQs. Answering queries in the presence of ontologies is a very natural problem. The input consists of an ontology, a database, and a query. One is interested in what the certain answers of the query are in the given database and under the given ontology. To transform the question into a decision problem, one can additionally give a candidate tuple $\mathbf{a}$ of constants from the database as an input and the question is whether $\mathbf{a}$ is a certain answer to the query.

Interestingly, this problem is already ExpTime-complete for many Horn DLs, which sounds like bad news for the usability of these logics in real-life knowledge representation scenarios. However, this result is slightly misleading because the complexity is usually measured relative to the size of the input and the database usually accounts for the biggest part of the input, while the query and the ontology are relatively small and often static. So there is a different, more refined way to measure the complexity, called *data complexity*: For every ontology $\mathcal{T}$ and query q, one considers the OMQ answering problem, where the input is only the database and a candidate tuple. Data complexity has been studied for many DLs [HMS05; KL07; Ros07; Cal+13; Bie+14; LW17], and measured in data complexity, answering Horn DL OMQs is usually tractable.

With this refined view on the complexity of answering OMQs, more questions arise. One can choose an ontology language $\mathcal{L}$ and a query language $\mathcal{Q}$ and ask: What are all the possible complexities of OMQs formulated in $\mathcal{L}$ and $\mathcal{Q}$? How can OMQs that belong to the same complexity class be characterized? And the so-called *meta problem*: How complex is it to decide what the complexity of a given OMQ is? See [LW12; Bie+14; ZKG18; LSW15] for initial results on these questions. To classify OMQs into different complexity classes, one is interested in results of the form 'every OMQ formulated in $\mathcal{L}$ and $\mathcal{Q}$ is either in complexity class X or hard for complexity class Y', which shows that there are no OMQs with a complexity that lies 'strictly between X and Y'. These so-called *dichotomy results* also play an important role in the complexity classification of *constraint satisfaction problems (CSP)*, the recently proven PTime/NP dichotomy (formerly known as the Feder-Vardi conjecture) being the most famous result from this area. And in fact, there is a very strong connection between complexities of CSP and the data complexity of OMQs [Bie+14].

Rewritability. How can OMQs be answered in practice? There are several reasoners, e.g. Protégé, Pellet and Fact++, that implement algorithms for reasoning with ontologies. However, traditional database management systems (DBMS) based on SQL or Datalog are still popular, since these have been developed for a long time and are nowadays highly optimized. This raises the question whether traditional DBMS can be utilized for answering OMQs, even though they do not explicitly provide this functionality. One way to achieve this is by *rewriting* the OMQ Q into an SQL or Datalog query q, which means to find a q such that the certain answers to Q are equal to the answers of q if executed on any database. It is not always possible to find such a rewriting q, since even for Horn DLs, rewritings into FO are not guaranteed to exist. But if a rewriting exists, one would certainly like to know this, to make use of the existing, very optimized DBMS.

So an interesting question is: Given an OMQ, is it rewritable into FO (as an abstraction of SQL) or into Datalog, or into some other relevant fragment of these? For results on rewritability of OMQs, see [Eit+12; BLW13; Bie+14; KNG14; FKL19]. In particular, we consider linear Datalog, the fragment of Datalog where recursive rules can only have one IDB atom in the body. We will see in Chapter 3 that rewritability into linear Datalog is strongly related to NL data complexity, a connection that has already been observed in [Dal05].

Query-by-example. Another reasoning problem we study is called *query-by-example (QBE)*. Imagine that someone explores a knowledge base and this person would like to formulate a query but is unable to do so since the person is unfamiliar with the ontology language or query language. However, the person can provide positive and negative examples from the data, i.e., data that should and data that should not be returned. The QBE problem asks: Is it possible to generalize the given examples into a query that returns at least all of the positive examples, but none of the given negative examples? In the positive case, we also want to compute said query. This problem is related to machine learning research: We want to learn a query from the given examples. QBE has been suggested in [Zlo75] and has been studied for traditional databases and different query languages [TCP14; CD15; BCS16; BR17; ADK16; BCL15]. We initiate the research on QBE for OMQs.

Expressiblility and verification. In ontology-based data access (OBDA), data from multiple sources is unified using a new, global vocabulary. The relations of the new vocabulary are defined in terms of the old vocabulary using queries (called *mappings*) over the data sources. Additionally, the global vocabulary is enriched with an ontology [Pog+08]. In the process of creating such an ontology, it might be unclear whether a certain query over the sources can be already expressed as a query over the global vocabulary, that is, whether there is an OMQ that when executed over the global vocabulary returns the same answers as the input query when executed over the old data sources. If this is not the case, introducing more mappings or changing the ontology might be necessary. So the expressibility problem asks, given an ontology, mappings, and a query q over the sources, whether q can be expressed as a query over the global vocabulary. The verification problem asks, additionally given a query q_t, whether q_t expresses q. We investigate these problems for several Horn DLs.

1.1 Structure of the Thesis

Besides the preliminaries (Chapter 2) and the conclusion (Chapter 6), this thesis has three main chapters.

In Chapter 3, we study the non-uniform data complexity of OMQs based on an $\mathcal{EL}$ ontology and a conjunctive query. We give a complete characterization regarding the data complexities of such OMQs, showing that for every such OMQ, the evaluation problem is either in AC^0 or NL-complete or PTime-complete. Additionally, we characterize rewritability into linear Datalog and show that it coincides with the OMQ having data complexity in NL and that there is no constant bound on the width of linear Datalog rewritings. We also investigate the meta problem to decide, given an OMQ, what its data complexity is and whether it is rewritable into linear Datalog, and show that this is ExpTime-complete.

In Chapter 4, we examine the QBE problem in the presence of ontologies. We focus on knowledge bases with Horn-$\mathcal{ALC}$ and $\mathcal{ELI}$ ontologies and show that the question of whether there exists a CQ that separates the positive from the negative examples is coNExpTime-complete for Horn-$\mathcal{ALC}$ and even undecidable for $\mathcal{ELI}$. Furthermore, we investigate the size of witness queries in the Horn-$\mathcal{ALC}$ case and show that there are cases of knowledge bases that require witness queries of double exponential size, and we show that double exponential size is always sufficient.

In Chapter 5, we study the expressibility and verification problem in the OBDA setting. We consider unions of conjunctive queries (UCQs) as source and target queries and global-as-view (GAV) mappings, showing that both problems are Π_2^p-complete in DL-Lite, coNExpTime-complete between $\mathcal{EL}$ and $\mathcal{ELHI}$ when source queries are rooted, and 2ExpTime-complete for unrestricted source queries.

1.2 Summary of Publications

Substantial parts of the results in this thesis have already been published in conference or workshop proceedings, or are currently under review. In detail:

Chapter 3

[LS17a] Carsten Lutz and Leif Sabellek. “Ontology-Mediated Querying with EL: Trichotomy and Linear Datalog Rewritability”. In: *Proceedings of DL-workshop*. 2017

[LS17b] Carsten Lutz and Leif Sabellek. “Ontology-Mediated Querying with the Description Logic EL: Trichotomy and Linear Datalog Rewritability”. In: *Proc. of IJCAI*. 2017, pp. 1181–1187

[LS19] Carsten Lutz and Leif Sabellek. “A Complete Classification of the Complexity and Rewritability of Ontology-Mediated Queries based on the Description Logic EL”. submitted to AI Journal. 2019

Chapter 4

[GJS18a] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. “Query-by-Example for Expressive Horn Description Logics”. In: *Proceedings of DL-workshop*. 2018

[GJS18b] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. “Reverse Engineering Queries in Ontology-Enriched Systems: The Case of Expressive Horn Description Logic Ontologies”. In: *Proceedings of IJCAI*. 2018, pp. 1847–1853

Chapter 5

[LMS18] Carsten Lutz, Johannes Marti, and Leif Sabellek. “Query Expressibility and Verification in Ontology-Based Data Access”. In: *Proceedings of KR*. AAAI Press, 2018, pp. 389–398

2 Preliminaries

In this chapter, we introduce the basic formalisms and tools used throughout the whole thesis. Additionally, a chapter might have its own preliminary section to introduce chapter-specific notions. We introduce first-order logic, the description logics that are relevant for this thesis, ontology-mediated queries, and we recall some well-known results about these topics.

2.1 First-order Logic

First-order logic (FO) is a fundamental logic used in mathematics and computer science. This thesis is concerned with many different fragments of FO, so even though we rarely speak about full first-order logic, we introduce it here to set the frame for the description logics and query languages introduced in the coming sections. We introduce a version of FO without function symbols and without constants.

Syntax

A *signature* is a set Σ of relational symbols, each symbol $R \in \Sigma$ being associated with a positive integer, called the *arity* of R. A symbol of arity k is called k-ary, symbols of arity 1 are called *unary*, symbols of arity 2 are called *binary*. An *FO formula* over Σ is build according to the following context free grammar:

$$\varphi ::= R(x_1, \dots, x_k) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x \varphi$$

where $R \in \Sigma$ is a k-ary relational symbol and $x_1, \dots, x_k, x$ are symbols from an infinite set of *variables*. An *FO(=) formula* is build according to the context free grammar from above, where the rule $\varphi ::= x = y$ is allowed as well, x, y being variables. By $\mathrm{var}(\varphi)$ we denote the set of all variables that appear in φ. The *free variables* of φ are defined as follows.

$$\begin{aligned}
\mathsf{free}(R(x_1, \dots, x_k)) &= \{x_1, \dots, x_k\} \\
\mathsf{free}(\neg\varphi) &= \mathsf{free}(\varphi) \\
\mathsf{free}(\varphi \wedge \psi) &= \mathsf{free}(\varphi) \cup \mathsf{free}(\psi) \\
\mathsf{free}(\exists x \varphi) &= \mathsf{free}(\varphi) \setminus \{x\}
\end{aligned}$$

When we write $\varphi(\mathbf{x})$, where $\mathbf{x}$ is a tuple of variables, we mean that φ is a first order formula where $\mathsf{free}(\varphi)$ are precisely the variables that occur in $\mathbf{x}$, ordered in some fixed linear order.

Semantics

The semantics of FO is defined in terms of relational structures. In fact, many other objects that we define later on (like databases, conjunctive queries, ABoxes and interpretations) can be regarded as relational structures.

A *relational structure over* Σ is a tuple $\mathfrak{A} = (A, \pi)$, where A is a non-empty set called the *domain* of $\mathfrak{A}$ and π is a function that assigns to every k-ary relational symbol $R \in \Sigma$ a k-ary relation over A. A *valuation* for $\mathfrak{A}$ is a function v that assigns to every variable an element from A. Truth of an FO formula $\varphi(x_1, \ldots, x_n)$ over Σ in $\mathfrak{A}$ under the valuation v is defined as follows:

$$\begin{array}{ll}
(\mathfrak{A}, v) \models R(x_1, \ldots, x_k) & \text{if } (v(x_1), \ldots, v(x_k)) \in \pi(R) \\
(\mathfrak{A}, v) \models \neg\varphi & \text{if not } (\mathfrak{A}, v) \models \varphi \\
(\mathfrak{A}, v) \models \varphi \wedge \psi & \text{if } (\mathfrak{A}, v) \models \varphi \text{ and } (\mathfrak{A}, v) \models \psi \\
(\mathfrak{A}, v) \models \exists x \varphi & \text{if there exists } a \in A \text{ such that } (\mathfrak{A}, v_{[x/a]}) \models \varphi \\
(\mathfrak{A}, v) \models x = y & \text{if } v(x) = v(y)
\end{array}$$

In the fourth line, $v_{[x/a]}$ is the valuation defined by $v_{[x/a]}(x) = a$ and $v_{[x/a]}(y) = v(y)$ for all other variables y. If $\mathbf{a} = (a_1, \ldots, a_n)$ is a tuple of elements from A and $\varphi(x_1, \ldots, x_n)$ is an FO formula, we write $\mathfrak{A} \models \varphi(\mathbf{a})$ if $(\mathfrak{A}, v) \models \varphi(\mathbf{x})$, where $v(x_i) = a_i$ for all $i \in \{1, \ldots, n\}$.

A Σ-*homomorphism* from a relational structure $\mathfrak{A} = (A, \pi_A)$ to a relational structure $\mathfrak{B} = (B, \pi_B)$ is a function $h : A \to B$ such that for every $R \in \Sigma$ and $\mathbf{a}$ a tuple of elements from A, $\mathbf{a} \in \pi_A(R)$ implies $h(\mathbf{a}) \in \pi_B(R)$. If Σ contains all symbols from the signature of $\mathfrak{A}$, then we drop the Σ and call h a *homomorphism*. If $\mathbf{a} = (a_1, \ldots, a_n)$ is a tuple from A and $\mathbf{b} = (b_1, \ldots, b_n)$ is a tuple from B, we write $h : (\mathfrak{A}, \mathbf{a}) \to (\mathfrak{B}, \mathbf{b})$ if h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ with $h(a_i) = b_i$ for all $i \in \{1, \ldots, n\}$. Since ABoxes, interpretations and conjunctive queries (defined later) can all be regarded as relational structures, they all inherit the notion of (Σ-)homomorphisms, also between objects of different kinds, like homomorphisms from ABoxes to interpretations or from conjunctive queries to interpretations.

2.2 Horn Description Logic TBoxes

Description Logics (DLs) are a large family of fragments of FO that are used for knowledge representation and reasoning. Unlike in FO, reasoning problems (such as satisfiability, validity, consequence, query answering) for DLs are usually decidable. A wide range of Description Logics has been considered in the literature, ranging from very small, inexpressive fragments of FO where the reasoning problems can be decided efficiently, up to very expressive DLs with higher computational complexity.

This thesis is only concerned with *Horn Description Logics*, which are on the lower end of the scale regarding both expressive power and complexity. Roughly stated, Horn DLs do not allow any kind of disjunction.

We introduce DL-Lite, $\mathcal{EL}$, $\mathcal{ELI}$, $\mathcal{ELHI}$, Horn-$\mathcal{ALC}$ and Horn-$\mathcal{ALCI}$. First, we define the syntax for all these DLs, and after that, we define the semantics. Throughout the whole thesis, let $\mathsf{N_C}$ and $\mathsf{N_R}$ be two fixed countably infinite sets of *concept names* and *role names*, respectively. A *role* is either a role name r or an *inverse role* r^- where r is a role name.

Syntax of the $\mathcal{EL}$ Family

An $\mathcal{EL}$*-concept* C is built according to the following context free grammar:

$$C ::= A \mid \top \mid C \sqcap C \mid \exists r.C$$

Here, A ranges over $\mathsf{N_C}$ and r ranges over $\mathsf{N_R}$. An $\mathcal{ELI}$*-concept* is built according to the same grammar, but with r ranging over roles instead of just roles names. (The $\mathcal{I}$ in $\mathcal{ELI}$ stands for 'inverse').

An $\mathcal{EL}$-TBox (resp. $\mathcal{ELI}$-TBox) is a finite set of *concept inclusions (CI)* of the form $C \sqsubseteq D$, where C and D are $\mathcal{EL}$-concepts (resp. $\mathcal{ELI}$-concepts).

A *role inclusion (RI)* takes the form $r \sqsubseteq s$ where r and s are roles. An $\mathcal{ELHI}$-TBox is a finite set of CIs and RIs, where the CIs form an $\mathcal{ELI}$-TBox.

Syntax of the DL-Lite Family

DLs from the DL-Lite family are specifically designed so that ontology-mediated queries (defined later) are always rewritable into FO. We only define one representative of the DL-Lite family, namely DL-Lite$^{\mathcal{R}}_{\mathsf{horn}}$. A DL-Lite$^{\mathcal{R}}_{\mathsf{horn}}$-ontology is a finite set of concept inclusions and role inclusions. The concept inclusions are of the form

$$B \sqcap \cdots \sqcap B \sqsubseteq B$$

where B is build according to the rule

$$B ::= A \mid \top \mid \bot \mid \exists r.\top \mid \exists r^-.\top$$

and A ranges over concept names. The role inclusions take one of the forms

$$r \sqsubseteq s \qquad r \sqsubseteq s^- \qquad r_1 \sqcap \cdots \sqcap r_n \sqsubseteq \bot$$

where $r, s, r_1, \ldots, r_n$ range over role names.

Syntax of the Horn-$\mathcal{ALC}$ Family

The Horn-$\mathcal{ALC}$ family, introduced in [KRH13], is the most expressive description logic we deal with in this thesis. Compared to the DL-Lite and the $\mathcal{EL}$ family, the syntax of the Horn-$\mathcal{ALC}$ family is more complex, which is due to the fact that it is the result of carefully extending the expressive power of the $\mathcal{EL}$ family with disjunctions and negations at the right places without losing important properties like the existence of universal models, introduced in Section 2.6.

To define the syntax of *Horn-$\mathcal{ALCI}$*, we use the following context free grammar:

$$L ::= \top \mid \bot \mid A \mid L \sqcap L \mid L \sqcup L \mid \exists r.L$$
$$R ::= \top \mid \bot \mid A \mid \neg A \mid R \sqcap R \mid \neg L \sqcup R \mid \exists r.R \mid \forall r.R$$

Again, A ranges over $\mathsf{N_C}$. A *Horn-$\mathcal{ALC}$-TBox* is a finite set of concept inclusions of the form $L \sqsubseteq R$, where r ranges over role names. If we let r range over roles instead of role names, a finite set of concept inclusions of the form $L \sqsubseteq R$ is a *Horn-$\mathcal{ALCI}$-TBox*.

The *size* of a TBox, a concept, or any other syntactic object O, denoted $|O|$, is the number of symbols needed to write O, with each concept and role name counting as one symbol.

Semantics

All concept inclusions from the DL-Lite family and the $\mathcal{EL}$ family are also concept inclusions in Horn-$\mathcal{ALCI}$. The semantics of Horn-$\mathcal{ALCI}$-TBoxes are given in terms of interpretations. An *interpretation* is a tuple $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set, called the *domain* of $\mathcal{I}$, and $\cdot^\mathcal{I}$ is a function that assigns to every concept name A a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ and to every role name r a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. The function $\cdot^\mathcal{I}$ can be inductively extended to assign to every Horn-$\mathcal{ALCI}$ concept a subset of $\Delta^\mathcal{I}$ in the following way.

$$\begin{aligned}
(C_1 \sqcap C_2)^\mathcal{I} &= C_1^\mathcal{I} \cap C_2^\mathcal{I} \\
(C_1 \sqcup C_2)^\mathcal{I} &= C_1^\mathcal{I} \cup C_2^\mathcal{I} \\
(\exists r.C)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \exists e \in \Delta^\mathcal{I} : (d,e) \in r^\mathcal{I} \wedge e \in C^\mathcal{I}\} \\
(\exists r^-.C)^\mathcal{I} &= \{e \in \Delta^\mathcal{I} \mid \exists d \in \Delta^\mathcal{I} : (d,e) \in r^\mathcal{I} \wedge d \in C^\mathcal{I}\} \\
(\forall r.C)^\mathcal{I} &= \{d \in \Delta^\mathcal{I} \mid \forall e \in \Delta^\mathcal{I} : (d,e) \in r^\mathcal{I} \rightarrow e \in C^\mathcal{I}\} \\
(\forall r^-.C)^\mathcal{I} &= \{e \in \Delta^\mathcal{I} \mid \forall d \in \Delta^\mathcal{I} : (d,e) \in r^\mathcal{I} \rightarrow d \in C^\mathcal{I}\} \\
(\neg C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies a role inclusion of the form $r \sqsubseteq s$ if $r^\mathcal{I} \subseteq s^\mathcal{I}$. It satisfies a role inclusion of the form $r_1 \sqcap \cdots \sqcap r_n \sqsubseteq \bot$ if $r_1^\mathcal{I} \cap \ldots \cap r_n^\mathcal{I} = \emptyset$. It satisfies a concept inclusion $C_1 \sqsubseteq C_2$ if $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$. An interpretation $\mathcal{I}$ is a *model of a TBox $\mathcal{T}$* if it satisfies all concept inclusions and all role inclusions from $\mathcal{T}$. If C_1 and C_2 are concepts, we write $\mathcal{T} \models C_1 \sqsubseteq C_2$ if for every model $\mathcal{I}$ of $\mathcal{T}$, we have $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$. A *$\mathcal{T}$-type t* is a set of concept names from $\mathcal{T}$ that is closed under $\mathcal{T}$-consequence, that is, if $\mathcal{T} \models \sqcap t \sqsubseteq A$, then $A \in t$. Every interpretation can be seen as a relational structure, so it inherits the notions of ($\Sigma-$)homomorphisms.

2.3 Databases, ABoxes and Knowledge Bases

Relational databases are the classical formalism to store data. In the context of databases, we call the signature Σ a *schema*. A Σ-*database* D is a set of *facts* $R(a_1, \ldots, a_n)$ where $R \in \Sigma$ is a relation name of arity n and $a_1, \ldots, a_n$ are from a fixed infinite set $\mathsf{N_I}$ of *constants*. We use $\mathsf{adom}(D)$ to denote the set of constants that occur in D. Every database can be seen as a relational structure over the domain $\mathsf{adom}(D)$ in a straightforward way.

An ABox is the standard formalism to store data in the field of DLs. ABoxes can be thought of as databases that have only unary and binary relations. A *DL-signature* is a signature using only symbols from $\mathsf{N_C} \cup \mathsf{N_R}$, where every $A \in \mathsf{N_C}$ is unary and every $r \in \mathsf{N_R}$ is binary. If Σ is a DL-signature, a Σ-*ABox* is a finite non-empty set of *concept assertions* of the form $A(a)$ and *role assertions* of the form $r(a, b)$, where $A \in \Sigma$ is a concept name, $r \in \Sigma$ is a role name and $a, b \in \mathsf{N_I}$ are constants, but we call them *individuals* in the context of ABoxes. The ABox signature plays the same role as a schema in the database literature [AHV95]. We write $\mathsf{ind}(\mathcal{A})$ for the set of all individuals that appear in some assertion in $\mathcal{A}$. Every ABox $\mathcal{A}$ can be seen as a relational structure over the domain $\mathsf{ind}(\mathcal{A})$ in a straightforward way and homomorphisms between ABoxes and interpretations are defined accordingly.

A pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ that consists of a TBox and an ABox is called a *knowledge base (KB)*. If the TBox is formulated in the description logic $\mathcal{L}$, then $\mathcal{K}$ is called an $\mathcal{L}$-*KB*.

An interpretation is a *model of an ABox* $\mathcal{A}$ if $\mathsf{ind}(\mathcal{A}) \subseteq \Delta^\mathcal{I}$, $a \in A^\mathcal{I}$ for every concept assertion $A(a) \in \mathcal{A}$ and $(a, b) \in r^\mathcal{I}$ for every role assertion $r(a, b) \in \mathcal{A}$. It is a *model of the knowledge base* $(\mathcal{T}, \mathcal{A})$ if it is a model of $\mathcal{T}$ and a model of $\mathcal{A}$ at the same time. We call a KB $(\mathcal{T}, \mathcal{A})$ *consistent* if $(\mathcal{T}, \mathcal{A})$ has a model. By $\mathsf{ind}(\mathcal{A}) \subseteq \Delta^\mathcal{I}$ we are adopting the so called *standard name assumption*, which also implies the *unique name assumption*. The latter means that in a model $\mathcal{I}$ of $\mathcal{A}$, it is not allowed that different individuals of $\mathcal{A}$ are identified to a single element in $\mathcal{I}$.

The *indegree* of an individual $a \in \mathsf{ind}(a)$ is the number of assertions $r(b, a) \in \mathcal{A}$ for some role name r and the *outdegree* is the number of assertions $r(a, b) \in \mathcal{A}$. The *degree* of a is the sum of its indegree and its outdegree. The *degree* of an ABox is the maximum degree of its individuals. Every ABox $\mathcal{A}$ is associated with a directed graph $G_\mathcal{A}$ with vertices $\mathsf{ind}(\mathcal{A})$ and edges $\{(a, b) \mid r(a, b) \in \mathcal{A}\}$. A directed graph G is a *tree* if it is acyclic, connected and has a unique vertex with indegree 0, which is then called the *root* of G. An ABox $\mathcal{A}$ is *tree-shaped* if $G_\mathcal{A}$ is a tree and there are no multi-edges, that is, $r(a, b) \in \mathcal{A}$ implies $s(a, b) \notin \mathcal{A}$ for all $s \neq r$ and $s(b, a) \notin \mathcal{A}$ for all role names s. The *root* of a tree-shaped ABox $\mathcal{A}$ is the root of $G_\mathcal{A}$ and we call an individual b a *descendant* of an individual a if $a \neq b$ and the unique simple path from the root to b contains a.

2.4 Query Languages

A *conjunctive query (CQ)* is an expression of the form

$$q(\mathbf{x}) \leftarrow R_1(\mathbf{y}_1) \land R_2(\mathbf{y}_2) \land \ldots \land R_n(\mathbf{y}_n)$$

where $\mathbf{x}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are tuples of variables, $R_1, \ldots, R_n$ are relational symbols and the length of $\mathbf{y}_i$ is the arity of R_i.[1] We call $q(\mathbf{x})$ the *head* of the CQ and $R_1(\mathbf{y}_1) \wedge \ldots \wedge R_n(\mathbf{y}_n)$ the *body*. The variables that appear in $\mathbf{x}$ are called *answer variables* of q and all other variables are called *quantified variables*. We further require the so-called safety condition: Every answer variable is required to appear in the body. We refer to a CQ just as q, or as $q(\mathbf{x})$ if we want to emphasize that $\mathbf{x}$ is the tuple of answer variables. By $\mathsf{var}(q)$ we denote the set of all variables that appear in (the head or the body of) q. The arity of q, denoted $\mathsf{ar}(q)$, is the length of $\mathbf{x}$ and q is *Boolean* if $\mathsf{ar}(q) = 0$. If $V \subseteq \mathsf{var}(q)$, then we use $q|_V$ to denote the restriction of q to the atoms that only use variables from V (this may change the arity of q). Abusing notation, we sometimes interpret q as the set of atoms in its body and write e.g. $R_1(\mathbf{y}_1) \in q$.

The safety-condition, that every answer variable has to appear in the body, will sometimes be relaxed for technical reasons, usually making proofs more uniform and obtaining stronger results. We introduce vocabulary to distinguish this kind of CQ. A CQ is called *unsafe* if not every answer variable appears in the body and it is called *safe* if every answer variable appears in the body. So unless stated otherwise, every CQ is safe.

We introduce some important subclasses of CQs. First, we need some vocabulary to describe the structure of a CQ. For every CQ q, one can consider the undirected graph G_q whose nodes are $\mathsf{var}(q)$ and the edge $\{x, y\}$ exists if and only if x and y appear together in an atom in the body q. We call q *connected* if G_q is a connected graph and we call q *rooted* if every connected component of G_q contains at least one answer variable. Every CQ q that formulated in a DL-signature can be viewed as an ABox $\mathcal{A}_q$ by viewing (answer and quantified) variables as individual names. A CQ q is *tree-shaped* if $\mathcal{A}_q$ is.

An *atomic query (AQ)* is a unary CQ of the form $q(x) \leftarrow A(x)$, where the body consists of a single unary atom $A(x)$. A *union of conjunctive queries (UCQ)* is a finite non-empty set of CQs that all have the same arity. A *potentially infinite UCQ* is a (not necessarily finite) non-empty set of CQs that all have the same arity. If q is a UCQ, the CQs in q are called the *disjuncts* of q. The arity of a UCQ is the arity of its CQs. A UCQ is *rooted* if every of its disjuncts is rooted.

We denote by AQ the set of all AQs, by CQ the set of all CQs, by rCQ the set of all rooted CQs, by conCQ the set of all connected CQs and by rUCQ the set of all rooted UCQs. Note that AQ $\subseteq$ rCQ $\subseteq$ CQ $\subseteq$ UCQ. For a given signature Σ and a query language $\mathcal{Q}$, we denote with $\mathcal{Q}_\Sigma$ the set of all queries in $\mathcal{Q}$ that use only names from Σ.

Every k-ary (U)CQ q defines a function, which we also call *query*, that takes a database D as an input and returns a relation $\mathsf{ans}_q(D) \subseteq \mathsf{adom}(D)^k$ in the following way. A *homomorphism from a CQ $q(\mathbf{x})$ to a database D* is a function $h : \mathsf{var}(q) \rightarrow \mathsf{adom}(D)$ such that $R(h(\mathbf{y})) \in D$ for every $R(\mathbf{y}) \in q$. A tuple $\mathbf{a} \in \mathsf{adom}(D)^{\mathsf{ar}(q)}$ is an *answer to q on D*, denoted $D \models q(\mathbf{a})$, if there is a *homomorphism* h from $q(\mathbf{x})$ to D with $h(\mathbf{x}) = \mathbf{a}$. We denote

[1]This definition of CQs, called the *rule based* definition [AHV95], is one of several equivalent ways to define CQs and it is the most suitable definition for the purposes of this thesis. These CQs have the same expressive power as CQs defined by an FO formula of the form $\exists \mathbf{y} \varphi(\mathbf{x}, \mathbf{y})$, where φ is a conjunction of relational atoms and equality atoms.

by $\mathsf{ans}_q(D)$ the set of all answers to q on D. If q is a UCQ, we define:

$$\mathsf{ans}_q(D) = \bigcup_{p \in q} \mathsf{ans}_p(D)$$

The analogue notions of $\mathcal{A} \models q(\mathbf{a})$, $\mathcal{I} \models q(\mathbf{a})$, $\mathsf{ans}_q(\mathcal{A})$ and $\mathsf{ans}_q(\mathcal{I})$ for ABoxes $\mathcal{A}$ and interpretations $\mathcal{I}$ are defined accordingly, by seeing the ABox or the interpretation as a database.

All queries we defined correspond to FO(=) formulas, and in fact, every FO(=) formula φ also defines a query: a tuple $\mathbf{a}$ is an answer to φ on D if and only if $D \models \varphi(\mathbf{a})$. We denote the set of all FO queries by FO and the set of all FO queries with equality by FO(=).

2.5 Ontology-Mediated Queries

An *ontology mediated query (OMQ)* is a triple $Q = (\mathcal{T}, \Sigma, q)$ of a TBox $\mathcal{T}$, a DL-signature Σ and a query q. If $\mathcal{A}$ is an ABox and $\mathbf{a}$ is a tuple over $\mathsf{ind}(\mathcal{A})$, we say that $\mathbf{a}$ is a *certain answer to* Q *on* $\mathcal{A}$, denoted $\mathcal{A} \models Q(\mathbf{a})$, if for every model $\mathcal{I}$ of the knowledge base $(\mathcal{T}, \mathcal{A})$, we have $\mathbf{a} \in \mathsf{ans}_q(\mathcal{I})$. If we want to emphasize which TBox is used in the OMQ, we write $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ instead of $\mathcal{A} \models Q(\mathbf{a})$. The set of all certain answers to Q on $\mathcal{A}$ is denoted as $\mathsf{cert}_Q(\mathcal{A})$.

If $\mathcal{L}$ is an ontology language (such as $\mathcal{EL}$, $\mathcal{ELI}$, ...) and $\boldsymbol{Q}$ is a query language (such as CQ, conCQ, UCQ, FO, ...), we define $(\mathcal{L}, \boldsymbol{Q})$ to be the set of all OMQs $Q = (\mathcal{T}, \Sigma, q)$ where $\mathcal{T}$ is formulated in $\mathcal{L}$, Σ is any DL-signature, and $q \in \boldsymbol{Q}$.

A *rewriting* of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ is a query q_r over Σ of the same arity as q such that for all Σ-ABoxes $\mathcal{A}$, $\mathsf{ans}_{q_r}(\mathcal{A}) = \mathsf{cert}_Q(\mathcal{A})$. We speak of a *FO rewriting* and call Q *FO rewritable* if $q_r \in \text{FO}(=)$. We speak of a *UCQ rewriting* if q_r is a UCQ, of an *infinitary UCQ rewriting* if q_r is a potentially infinite UCQ. Basically all OMQs that we are concerned with in this thesis lie in (FO, UCQ) and we remark that for these OMQs, there always exists a *canonical infinitary UCQ rewriting* that is obtained by taking all Σ-ABoxes $\mathcal{A}$ and answers $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ and including $(\mathcal{A}, \mathbf{a})$ viewed as a CQ as a disjunct, that is, the assertions in $\mathcal{A}$ become the body of the rule and $\mathbf{a}$ the tuple of answer variables, using individuals as variables. In fact, this follows from the definition of rewritings and the fact that OMQs with $\mathcal{T}$ formulated in FO without equality are preserved under homomorphisms [Bie+14].

Parts of this thesis are concerned with the complexity of evaluating OMQs. One way to look at this problem is to receive as input an OMQ Q, an ABox $\mathcal{A}$ and a tuple $\mathbf{a}$ from $\mathsf{ind}(\mathcal{A})$ and ask whether $\mathcal{A} \models Q(\mathbf{a})$. However, defining the problem in this way gives only an approximate impression on the complexity of the problem, because the complexity is usually measured relative to the size of the input, and the database usually accounts for the biggest part of the input, while the query and the ontology are relatively small and often static. So there is a different, more refined way to measure the complexity, called *data complexity* as opposed to the former, which is called *combined complexity*. For every OMQ $Q = (\mathcal{T}, \Sigma, q)$ we define the problem $\textsc{eval}(Q)$ as follows:

	EVAL(Q)
Input:	A Σ-ABox $\mathcal{A}$ and a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$
Question:	$\mathcal{A} \models Q(\mathbf{a})$?

2.6 Universal Models

In Horn-DLs, data complexity of the evaluation problem is usually lower compared to their non-Horn counterparts. This is closely related to the fact that for every knowledge base $(\mathcal{T}, \mathcal{A})$ with $\mathcal{T}$ formulated in a Horn-DL such as Horn-$\mathcal{ALCI}$, there exists an interpretation that is a model for $(\mathcal{T}, \mathcal{A})$ and that has several nice properties, as summarized by the following lemma:

Lemma 2.1. *Let $\mathcal{T}$ be an Horn-$\mathcal{ALCI}$-TBox and $\mathcal{A}$ an ABox. Then there is an interpretation $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ such that*

1. *$\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is a model of $\mathcal{A}$ and $\mathcal{T}$;*
2. *for every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$, there is a homomorphism from $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to $\mathcal{I}$ that is the identity on $\mathsf{ind}(\mathcal{A})$;*
3. *for all UCQs q and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$ if and only if $\mathcal{U}_{\mathcal{A},\mathcal{T}} \models q(\mathbf{a})$.*

We present two well known ways to construct a model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. One is based on a standard chase procedure, which constructs $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ as the limit of a process of extending $\mathcal{A}$ with new assertions. The other way is a direct construction, which is sometimes more convenient to use. Note however that point 1 and 2 of Lemma 2.1 imply that each two universal models of a knowledge base are homomorphically equivalent.

TBox Normal Form

For both constructions, we assume that the TBox is in a *normal form*, which means it only uses concept inclusions of a certain restricted form. An $\mathcal{ELHI}$-TBox is in normal form if every of its concept inclusions takes one of the forms

$$\exists r.A_1 \sqsubseteq A_2 \qquad \top \sqsubseteq A_1 \qquad A_1 \sqcap A_2 \sqsubseteq A_3 \qquad A_1 \sqsubseteq \exists r.A_2\,,$$

where all A_i are concept names and r is a role. An $\mathcal{EL}$-TBox is in normal form if it only uses concept inclusions of this form, but with r a role name. A Horn-$\mathcal{ALCI}$-TBox is in normal form if every of its concept inclusions is of one of these forms or the form $A_1 \sqsubseteq \bot$.

It is well-known that every $\mathcal{EL}$ (resp. $\mathcal{ELHI}$, Horn-$\mathcal{ALCI}$) ontology $\mathcal{T}$ can be converted into an $\mathcal{EL}$ (resp. $\mathcal{ELHI}$, Horn-$\mathcal{ALCI}$) ontology $\mathcal{T}'$ in normal form in linear time [BBL05; Baa+17; Eit+08; Bie+16]. This process introduces new concept names. The resulting TBox $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$, that is, every model of $\mathcal{T}'$ is a model of $\mathcal{T}$ and, conversely, every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by interpreting the fresh concept names. Consequently, when $\mathcal{T}$ is replaced in an OMQ $Q = (\mathcal{T}, \Sigma, q)$ with $\mathcal{T}'$, resulting in an OMQ Q', then Q and Q' are equivalent in the sense that they give the same answers on all Σ-ABoxes. Thus, conversion of the TBox in an OMQ into normal form does not impact its data complexity nor rewritability into any other query language. We use TBox normal form in many proofs throughout the thesis.

Universal Model for $\mathcal{ELHI}$ Using the Chase Procedure

We give a construction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ for $\mathcal{ELHI}$-TBoxes using a (lazy) chase procedure.[2] Thus, let $\mathcal{T}$ be an $\mathcal{ELHI}$-TBox in normal form and $\mathcal{A}$ an ABox. It is convenient to use ABox notation for the construction, so we define an infinite sequence of ABoxes $\mathcal{A}_0 \subseteq \mathcal{A}_1 \subseteq \dots$ and define $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to be the interpretation corresponding to the limit $\bigcup_{i=0}^{\infty} \mathcal{A}_i$. We set $\mathcal{A}_0 = \mathcal{A}$ and then let $\mathcal{A}_{i+1}$ be $\mathcal{A}_i$ extended as follows:

(i) If $\exists r.B \sqsubseteq A \in \mathcal{T}$, $r(a,b), B(b) \in \mathcal{A}_i$ and $A(a) \notin \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_{i+1}$.

(ii) If $\exists r^-.A \sqsubseteq B \in \mathcal{T}$, $r(a,b), A(a) \in \mathcal{A}_i$ and $B(b) \notin \mathcal{A}_i$, then add $B(b)$ to $\mathcal{A}_{i+1}$.

(iii) If $\top \sqsubseteq A \in \mathcal{T}$ and $a \in \mathsf{ind}(\mathcal{A}_i)$, then add $A(a)$ to $\mathcal{A}_{i+1}$.

(iv) If $B_1 \sqcap B_2 \sqsubseteq A \in \mathcal{T}$ and $B_1(a), B_2(a) \in \mathcal{A}_i$, then add $A(a)$ to $\mathcal{A}_{i+1}$.

(v) If $A \sqsubseteq \exists r.B \in \mathcal{T}$, $A(a) \in \mathcal{A}_i$ and there is no $b \in \mathsf{ind}(\mathcal{A}_i)$ such that $r(a,b)$ and $B(b)$, then take a fresh individual b and add $r(a,b)$ and $B(b)$ to $\mathcal{A}_{i+1}$.

(vi) If $B \sqsubseteq \exists r^-.A \in \mathcal{T}$, $B(b) \in \mathcal{A}_i$ and there is no $a \in \mathsf{ind}(\mathcal{A}_i)$ such that $r(a,b)$ and $A(a)$, then take a fresh individual a and add $r(a,b)$ and $A(a)$ to $\mathcal{A}_{i+1}$.

(vii) If $r \sqsubseteq s \in \mathcal{T}$ and $r(a,b) \in \mathcal{A}_i$, then add $s(a,b)$ to $\mathcal{A}_{i+1}$.

Let $\mathcal{A}_\omega = \bigcup_{i \geq 0} \mathcal{A}_i$ and define $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ to be $\mathcal{A}_\omega$ seen as an interpretation. This does actually not define $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ in a unique way since the order of applying the above rules may have an impact on the shape of $\mathcal{A}_\omega$. However, all resulting $\mathcal{A}_\omega$ are homomorphically equivalent and it does not matter for the constructions in this thesis which order we use. Slightly sloppily, we thus live with the fact that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is not uniquely defined. Note that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can be infinite and that its shape is basically the shape of $\mathcal{A}$, but with a (potentially infinite) tree attached to every individual in $\mathcal{A}$. The domain elements in these trees are introduced by Rules (v) and (vi), and we refer to them as *anonymous elements*. The properties in Lemma 2.1 are standard to prove, see for example [BO15; Baa+17] for similar proofs.

Using this construction of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, we can also prove the following lemma, which allows us to concentrate on ABoxes of small degree.

Lemma 2.2. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELHI}, \mathrm{UCQ})$ be an OMQ and $\mathcal{A}$ a Σ-ABox such that $\mathcal{A} \models Q(\mathbf{a})$. Then there exists $\mathcal{A}' \subseteq \mathcal{A}$ of degree at most $|\mathcal{T}|$ such that $\mathcal{A}' \models Q(\mathbf{a})$.*

[2] This version of the chase is called lazy, since rules (v) and (vi) are only applied when the consequence is not satisfied. This is in contrast to the *oblivious* chase, where these rules are also applied when the consequence is already satisfied. The reason we use the lazy chase is that the oblivious chase produces a model with infinite outdegree, which is sometimes inconvenient for technical reasons.

Proof. (sketch) Assume $\mathcal{A} \models Q(\mathbf{a})$ and let $\mathcal{A}_\omega$ be the ABox produced by the chase procedure described above. Since $\mathcal{A} \models Q(\mathbf{a})$, by Lemma 2.1, $A(\mathbf{a}) \in \mathcal{A}_\omega$. Let $\mathcal{A}'$ be obtained from $\mathcal{A}$ by removing all assertions $r(a, b)$ that did not participate in any application of rule (i), (ii), (v) or (vi) and let $\mathcal{A}'_c$ be the result of chasing $\mathcal{A}'$. Clearly, we must have $A(\mathbf{a}) \in \mathcal{A}'_c$. Moreover, it is easy to verify that the degree of $\mathcal{A}'$ is at most $|\mathcal{T}|$. □

Universal Model for Horn-$\mathcal{ALCI}$ Using a Direct Construction

For Horn-$\mathcal{ALCI}$-TBoxes, we give a direct construction for a model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that satisfies the properties from Lemma 2.1. This direct construction is more convenient to use in some proofs in Chapter 4. Let $(\mathcal{T}, \mathcal{A})$ be a consistent Horn-$\mathcal{ALCI}$ KB and $\mathcal{T}$ in normal form. Assume w.l.o.g. that every concept or role name from $\mathcal{A}$ also appears in $\mathcal{T}$. When $a \in \mathsf{ind}(\mathcal{A})$, t, t' are types for $\mathcal{T}$, and r is a role, we write

- $a \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} t$ if $\mathcal{T}, \mathcal{A} \models \exists r. \bigsqcap t(a)$ and t is $\subseteq$-maximal with this condition, and
- $t \rightsquigarrow_r^{\mathcal{T}} t'$ if $\mathcal{T} \models \bigsqcap t \sqsubseteq \exists r. \bigsqcap t'$ and t' is $\subseteq$-maximal with this condition.

A *path for $\mathcal{A}$ and $\mathcal{T}$* is a finite sequence $\pi = ar_0t_1 \cdots t_{n-1}r_{n-1}t_n$, $n \geq 0$, with $a \in \mathsf{ind}(\mathcal{A})$, $r_0, \ldots, r_{n-1}$ roles, and $t_1, \ldots, t_n$ types for $\mathcal{T}$ such that

(i) $a \rightsquigarrow_{r_0}^{\mathcal{T},\mathcal{A}} t_1$ and (ii) $t_i \rightsquigarrow_{r_i}^{\mathcal{T}} t_{i+1}$ for every $1 \leq i < n$.

We use $\mathsf{tail}(\pi)$ to denote the last element of a path π. Let Paths be the set of all paths for $\mathcal{A}$ and $\mathcal{T}$ and note that paths with $n = 0$ correspond to $\mathsf{ind}(\mathcal{A})$. The universal model $\mathcal{U}_{\mathcal{T},\mathcal{A}}$ of $(\mathcal{T}, \mathcal{A})$ is defined as follows:

$$\begin{aligned}
\Delta^{\mathcal{U}_{\mathcal{T},\mathcal{A}}} &= \mathsf{Paths} \\
A^{\mathcal{U}_{\mathcal{T},\mathcal{A}}} &= \{a \in \mathsf{ind}(\mathcal{A}) \mid \mathcal{T}, \mathcal{A} \models A(a)\} \cup \\
&\quad\ \{\pi \in \mathsf{Paths} \setminus \mathsf{ind}(\mathcal{A}) \mid A \in \mathsf{tail}(\pi)\} \\
r^{\mathcal{U}_{\mathcal{T},\mathcal{A}}} &= \{(a, b) \in \mathsf{ind}(\mathcal{A})^2 \mid r(a, b) \in \mathcal{A}\} \cup \\
&\quad\ \{(\pi, \pi r t) \mid \pi r t \in \mathsf{Paths}\} \cup \\
&\quad\ \{(\pi r^- t, \pi) \mid \pi r^- t \in \mathsf{Paths}\}
\end{aligned}$$

It is well-known that the resulting interpretation $\mathcal{U}_{\mathcal{T},\mathcal{A}}$ is a universal model [BO15].

2.7 Pseudo Tree-Shaped ABoxes

Throughout the thesis, we often concentrate on ABoxes that take a restricted, almost tree-shaped form. These are called pseudo tree-shaped ABoxes, introduced in [Bie+16]. An ABox $\mathcal{A}$ is a *pseudo tree-shaped ABox of core size n* if there exist ABoxes $C, \mathcal{A}_1, \ldots, \mathcal{A}_k$ such that $\mathcal{A} = C \cup \bigcup_{i=1}^{k} \mathcal{A}_i$, $|\mathsf{ind}(C)| = n$, and all $\mathcal{A}_i$ are tree-shaped ABoxes with pairwise disjoint individuals and $\mathsf{ind}(C) \cap \mathsf{ind}(\mathcal{A}_i)$ consists precisely of the root of $\mathcal{A}_i$. We call C the *core* of $\mathcal{A}$. The tree-shaped ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_k$ that are part of a pseudo tree-shaped ABox should not be confused with the anonymous trees that are added when chasing a pseudo tree-shaped ABox to construct a universal model. Note that every tree-shaped ABox is pseudo tree-shaped with core size 1.

The following lemma, which is an adaptation of Proposition 23 in Appendix B of [Bie+16], describes the central property of pseudo tree-shaped ABoxes. It essentially says that if $\mathbf{a}$ is an answer to an OMQ Q based on a connected CQ q on an ABox $\mathcal{A}$, then one can unravel $\mathcal{A}$ into a pseudo tree-shaped ABox $\mathcal{A}'$ that homomorphically maps to $\mathcal{A}$ and such that $\mathbf{a}$ is an answer to Q on $\mathcal{A}'$, witnessed by a homomorphism from q to $\mathcal{U}_{\mathcal{A}',\mathcal{T}}$ that satisfies the additional property of being within or at least 'close to' the core of $\mathcal{A}'$.

Lemma 2.3. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELHI}, UCQ)$ an OMQ, $\mathcal{A}$ an ABox and $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$. Then there is a pseudo tree-shaped Σ-ABox $\mathcal{A}'$ of core size at most $|q|$ and a tuple $\mathbf{a}'$ in the core of $\mathcal{A}'$ such that*

1. *the degree of $\mathcal{A}'$ is not larger than $|\mathcal{T}|$;*
2. *$\mathbf{a}' \in \mathsf{cert}_Q(\mathcal{A}')$;*
3. *there is a homomorphism h from $\mathcal{A}'$ to $\mathcal{A}$ with $h(\mathbf{a}') = \mathbf{a}$.*

The statement remains true if Condition 2 is replaced with the following Condition $2'$, where C' denotes the core of $\mathcal{A}'$.

$2'$. $\mathbf{a}' \in \mathsf{cert}_Q(C' \cup \{A(a) \mid \mathcal{A}', \mathcal{T} \models A(a),\ a \in \mathsf{ind}(C')\})$.

Condition $2'$, which is strictly stronger than Condition 2, essentially says that there is a homomorphism h from a disjunct of q to the universal model of $\mathcal{A}'$ and $\mathcal{T}$ that only involves elements from the core and anonymous elements below them.

Proof. (sketch) Assume that $\mathcal{A} \models Q(\mathbf{a})$. By Lemma 2.1, there is a homomorphism g from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $g(\mathbf{x}) = \mathbf{a}$. Let $I \subseteq \mathsf{ind}(\mathcal{A})$ be the set of all individuals b that are either in the range of g or such that an anonymous element in the chase-generated tree below b is in the range of h. We can unravel $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ into a potentially infinite pseudo tree-shaped ABox $\mathcal{A}_0$ with core I, see [Bie+16] for details. Then $\mathcal{A}_0 \models Q(\mathbf{a})$ and this is witnessed by a homomorphism required for Condition $2'$. However, $\mathcal{A}_0$ does not have to be finite. By the compactness theorem of first order logic, there exists a finite subset $\mathcal{A}_1 \subseteq \mathcal{A}_0$ such that $\mathcal{A}_1 \models Q(\mathbf{a})$. Let $\mathcal{A}'$ be the restriction of $\mathcal{A}_1$ to those individuals that are reachable in $G_{\mathcal{A}'}$ from an individual in I. It can be verified that $\mathcal{A}'$ is as required. □

2.8 Derivation Trees

Entailment of AQs under TBoxes formulated in Horn DLs can be characterized in terms of so-called derivation trees [BLW13]. A derivation tree is a tree-shaped structure that encodes a proof of a statement of the form $\mathcal{A}, \mathcal{T} \models A(a)$. Several decision procedures in this thesis rely on the following characterization.

Let $\mathcal{T}$ be an $\mathcal{ELHI}$-TBox in normal form and $\mathcal{A}$ and ABox, $A_0 \in \mathsf{N_C}$ and $a_0 \in \mathsf{ind}(\mathcal{A})$. A *derivation tree* for a fact $A_0(a_0)$ in $\mathcal{A}$ is a finite $\mathsf{ind}(\mathcal{A}) \times \mathsf{N_C}$-labelled tree (T, V) that satisfies the following conditions:

1. $V(\varepsilon) = (a_0, A_0)$;

2. if $V(x) = (a, A)$, then one of the following holds:
   - x is a leaf and additionally, $A(a) \in \mathcal{A}$ or $\top \sqsubseteq A \in \mathcal{T}$;
   - x has successors $y_1, \dots, y_k$, $k \geq 1$ with $V(y_i) = (a, B_i)$ for $1 \leq i \leq k$ and $\mathcal{T} \models B_1 \sqcap \dots \sqcap B_k \sqsubseteq A$;
   - x has a single successor y with $V(y) = (b, B)$ and there is an $\exists s.B \sqsubseteq A \in \mathcal{T}$ and an $r(a, b) \in \mathcal{A}$ such that $\mathcal{T} \models r \sqsubseteq s$.

Note that the second item of Point 2 above requires $\mathcal{T} \models A_1 \sqcap \dots \sqcap A_n \sqsubseteq A$ instead of $A_1 \sqcap A_2 \sqsubseteq A \in \mathcal{T}$ to 'shortcut' anonymous parts of the universal model. In fact, the derivation of A from $A_1 \sqcap \dots \sqcap A_n$ by $\mathcal{T}$ can involve the introduction of anonymous elements. The following lemma is proved in [Bie+16] for the stronger logic $\mathcal{ELIHF}_\bot^{\sqcap-\mathsf{lhs}}$, whereas we only need it for $\mathcal{ELHI}$.

Lemma 2.4. *Let $\mathcal{T}$ be a $\mathcal{ELHI}$-TBox, $\mathcal{A}$ and ABox, $A \in \mathsf{N_C}$ and $a \in \mathsf{ind}(\mathcal{A})$. Then $\mathcal{A}, \mathcal{T} \models A(a)$ if and only if there is a derivation tree for $A(a)$ in $\mathcal{A}$.*

2.9 Two-way Alternating Parity Automata (TWAPA)

Tree automata are a versatile tool that we use throughout the thesis to obtain upper bounds for the complexities of different decision problems. A tree automaton receives a labelled tree as an input, performs some algorithmic steps on it, and finally accepts or rejects the tree. We usually encode (pseudo) tree-shaped ABoxes as labelled trees and use tree automata for questions like whether a certain concept name is derived at a certain individual, whether an OMQ is entailed, and much more complicated questions. The kind of tree automata we use are *two-way alternating parity automata on finite trees (TWAPA)*, introduced in [Var98].

Definition of TWAPA

Let $\mathbb{N} = \{1, 2, 3, \dots\}$. A *tree* is a non-empty (and potentially infinite) set $T \subseteq \mathbb{N}^*$ closed under prefixes.[3] We say that T is *m-ary* if $T \subseteq \{1, \dots, m\}^*$. For an alphabet Γ, a *Γ-labelled tree* is a pair (T, L) with T a tree and $L : T \to \Gamma$ a node labelling function. An *infinite path* P of a tree T is a prefix-closed set $P \subseteq T$ such that for every $i \geq 0$, there is a unique $x \in P$ with $|x| = i$.

For any set X, let $\mathcal{B}^+(X)$ denote the set of all positive Boolean formulas over X, i.e., formulas built using conjunction and disjunction over the elements of X used as propositional variables, and where the special formulas true and false are allowed as well.

Definition 2.5 (TWAPA)**.** *A* two-way alternating parity automaton (TWAPA) on finite *m-ary trees is a tuple $\mathfrak{A} = (S, \Gamma, \delta, s_0, c)$ where S is a finite set of* states, *Γ is a finite alphabet,*

[3]The * is the Kleene star, so $\mathbb{N}^*$ are all finite sequences of natural numbers, including the sequence of length 0, denoted ε.

$\delta : S \times \Gamma \longrightarrow \mathcal{B}^+(\mathsf{tran}(\mathfrak{A}))$ *is the* transition function *with* $\mathsf{tran}(\mathfrak{A}) = \{\langle i \rangle s,\ [i]s \mid -1 \leq i \leq m \text{ and } s \in S\}$ *the set of* transitions *of* $\mathfrak{A}$, $s_0 \in S$ *is the* initial state, *and* $c : S \longrightarrow \mathbb{N}$ *is the* parity condition *that assigns to each state a* priority.

Intuitively, a transition $\langle i \rangle s$ with $i > 0$ means that a copy of the automaton in state s is sent to the i-th successor of the current node, which is then required to exist. Similarly, $\langle 0 \rangle s$ means that the automaton stays at the current node and switches to state s, and $\langle -1 \rangle s$ indicates moving to the predecessor of the current node, which is then required to exist. Transitions $[i]s$ mean that a copy of the automaton in state s is sent on the relevant successor if that successor exists (which is not required).

Definition 2.6 (Run, Acceptance)**.** *Let* $\mathfrak{A} = (S, \Gamma, \delta, s_0, c)$ *be a TWAPA and* (T, L) *a finite* Γ*-labelled tree. A* configuration *is a pair from* $T \times S$*. A* run *of* $\mathfrak{A}$ *on* (T, L) *from the configuration* γ *is a* $T \times S$*-labelled tree* (T_r, r) *such that the following conditions are satisfied:*

1. $r(\varepsilon) = \gamma$*;*
2. *if* $y \in T_r$, $r(y) = (x, s)$, *and* $\delta(s, L(x)) = \varphi$, *then there is a (possibly empty) set* $S \subseteq \mathsf{tran}(\mathfrak{A})$ *such that* S *(viewed as a propositional valuation) satisfies* φ *as well as the following conditions:*
   a) *if* $\langle i \rangle s' \in S$, *then* $x \cdot i \in T$ *and there is a node* $y \cdot j \in T_r$ *such that* $r(y \cdot j) = (x \cdot i, s')$*;*
   b) *if* $[i]s' \in S$ *and* $x \cdot i \in T$, *then there is a node* $y \cdot j \in T_r$ *such that* $r(y \cdot j) = (x \cdot i, s')$*.*

We say that (T_r, r) *is* accepting *if on all infinite paths of* T_r*, the maximum priority that appears infinitely often on this path is even. A finite* Γ*-labelled tree* (T, L) *is* accepted *by* $\mathfrak{A}$ *if there is an accepting run of* $\mathfrak{A}$ *on* (T, L) *from the configuration* (ε, s_0)*. We use* $L(\mathfrak{A})$ *to denote the set of all finite* Γ*-labelled tree accepted by* $\mathfrak{A}$*.*

It is known (and easy to see) that TWAPAs are closed under complementation and intersection, and that these constructions involve only a polynomial blowup. In particular, complementation boils down to dualizing the transitions and increasing all priorities by one. The *emptiness problem of TWAPA* is the problem to decide whether a given TWAPA accepts at least one tree. The following lemma is implicit in [Var98].

Lemma 2.7. *The emptiness problem of TWAPA can be solved in time single exponential in the number of states and in the highest occurring priority, and polynomial in all other components of the TWAPA.*

This lemma and the fact that all TWAPA we construct only use priorities 1 and 2, allow us to only explicitly analyse the number of states, but only implicitly take care that all other components are of the allowed size for the complexity result that we aim to obtain.

2.10 Computational Complexity

The largest part of this thesis is concerned with pinpointing the computational complexity of reasoning problems. The following figure gives an overview of all relevant complexity classes that appear in this thesis.

$$\begin{tikzcd}
& \mathrm{NP} \arrow[dr, phantom, sloped, "\subseteq"] & & \textsc{NExpTime} \arrow[dr, phantom, sloped, "\subseteq"] & \\
\mathrm{AC}^0 \subsetneq \textsc{LogSpace} \subseteq \mathrm{NL} \subseteq \textsc{PTime} \arrow[ur, phantom, sloped, "\subseteq"] \arrow[dr, phantom, sloped, "\subseteq"] & & \Pi_2^p \subseteq \textsc{PSpace} \subseteq \textsc{ExpTime} \arrow[ur, phantom, sloped, "\subseteq"] \arrow[dr, phantom, sloped, "\subseteq"] & & 2\text{-}\textsc{ExpTime} \\
& \textsc{co}\mathrm{NP} \arrow[ur, phantom, sloped, "\subseteq"] & & \textsc{coNExpTime} \arrow[ur, phantom, sloped, "\subseteq"] &
\end{tikzcd}$$

The least familiar classes here might be AC^0 and Π_2^p. The class AC^0 is defined to be the set of all problems that can be solved using a family of Boolean circuits of polynomial size and logarithmic depth, where the fan-in of AND-gates and OR-gates is not restricted. The significance of AC^0 comes from the fact that the evaluation problem of a fixed FO formula on a given input structure lies in AC^0. Thus, if an OMQ is FO rewritable, its data complexity lies in AC^0. This class plays a role in Chapter 3, where we classify the data complexity of OMQs from $(\mathcal{EL}, \mathrm{CQ})$.

The class Π_2^p is situated in the second level of the polynomial hierarchy and can be alternatively defined as coNP$^{\mathrm{NP}}$, the class of problems solvable by a coNP-machine using an oracle for an NP-complete problem, see [AB09]. A complete problem for this class is $\forall\exists$SAT, which asks whether a given formula of the form

$$\varphi = \forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m \; \psi(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

is true, where ψ is a formula in propositional logic. In Chapter 5, we encounter a problem that can encode $\forall\exists$SAT and turns out to be Π_2^p-complete.

We remind the reader that completeness for all relevant complexity classes above PTime is usually defined in terms of PTime many-one reductions. For classes below PTime (including the class PTime itself), one defines completeness via LogSpace many-one reductions. However, whenever possible, we will prove completeness results under the even weaker notion of FO reductions. Roughly said, an FO reduction is a many-one reduction given by a single FO formula, which defines the output structure in terms of the input structure, see [Imm99] for more details. Every FO reduction is also a LogSpace reduction, so if we prove hardness under FO reductions, this is a stronger result than proving hardness under LogSpace reductions.

3 A complete classification of complexity and rewritability for $(\mathcal{EL}, \mathrm{CQ})$

Prominent choices for ontology languages in OMQs include DLs like $\mathcal{EL}$, Horn-$\mathcal{SHIQ}$, and $\mathcal{ALC}$ [Baa+17], while the most common choices for query languages in OMQs are CQs, UCQs and AQs. Substantial research efforts have been invested into understanding the properties of the resulting OMQ languages, with two important topics being

1. the *data complexity* of OMQ evaluation [HMS05; KL07; Ros07; Cal+13; Bie+14; LW17], where data complexity means that only the data is considered the input while the OMQ is fixed, and

2. the *rewritability* of OMQs into more standard database query languages such as SQL (which in this context is often equated with first-order logic) and Datalog [Eit+12; BLW13; Bie+14; KNG14; FKL19].

PTime data complexity is often considered a necessary condition for efficient query evaluation in practice. Questions about rewritability are also motivated by practical concerns: Since most database systems are unaware of ontologies, rewriting OMQs into standard database query languages provides an important avenue for implementing OMQ execution in practical applications [Cal+09; Han+15; PMH10; Tri+15]. Both subjects are thoroughly intertwined since rewritability into first-order logic (FO) is closely related to AC^0 data complexity while rewritability into Datalog is closely related to PTime data complexity. We remark that FO rewritability of an OMQ implies rewritability into a UCQ and thus into Datalog [Bie+14]. In this chapter, when speaking about complexity we always mean data complexity.

Regarding compexity and rewritability, modern DLs can roughly be divided into two families: 'expressive DLs' such as $\mathcal{ALC}$ and $\mathcal{SHIQ}$ that result in OMQ languages with coNP complexity and where rewritability is guaranteed neither into FO nor into Datalog [Bie+14; Tri+15; FKL19], and 'Horn DLs' such as $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ which typically have PTime complexity and where rewritability into (monadic) Datalog is guaranteed, but FO rewritability is not [BLW13; Han+15; Bie+16]. In practical applications, however, ontology engineers often need to use language features that are only available in expressive DLs, but they typically do so in a way such that one may hope for hardness to be avoided by the concrete ontologies that are being designed.

Initiated in [LW12; Bie+14], this has led to studies of data complexity and rewritability that are much more fine-grained than the analysis of entire ontology languages, see also [ZKG18; LSW15]. The ultimate aim is to understand, for relevant OMQ languages $(\mathcal{L}, \mathcal{Q})$, the exact complexity and rewritability status of every OMQ from $(\mathcal{L}, \mathcal{Q})$. For expressive DLs, this turns out to be closely related to the complexity classification of constraint satisfaction problems (CSPs) with a fixed template [FV98; All+09]. Very important progress has recently been made in this area with the proof that CSPs enjoy a dichotomy between PTime and NP [Bul17; Zhu17]. Via the results in [Bie+14], this implies that OMQ evaluation in languages such as $(\mathcal{ALC}, \text{UCQ})$ enjoys a dichotomy between PTime and coNP. However, the picture is still far from being fully understood. For example, neither in CSP nor in expressive OMQ languages it is known whether there is a dichotomy between NL and PTime, and whether containment in NL coincides with rewritability into linear Datalog [Dal05].

We remark that rewritability into linear Datalog might also be interesting from a practical perspective. In fact, the equation "SQL = FO" often adopted in ontology-mediated querying ignores the fact that SQL contains linear recursion from its version 3 published in 1999 on, which exceeds the expressive power of FO. We believe that, in the context of OMQs, linear Datalog might be a natural abstraction of SQL that includes linear recursion, despite the fact that it does not contain full FO. Indeed, the fact that all OMQs from $(\mathcal{EL}, \text{CQ})$ that are FO rewritable are also UCQ-rewritable shows that the expressive power of FO that lies outside of linear Datalog is not useful when using SQL as a target language for OMQ rewriting.

Contribution and Structure of the Chapter

The aim of this chapter is to carry out an ultimately fine-grained analysis of the data complexity and rewritability of OMQs from the languages $(\mathcal{EL}, \text{CQ})$ and $(\mathcal{EL}, \text{AQ})$. In fact, we completely settle the complexity and rewritability status of each OMQ from $(\mathcal{EL}, \text{CQ})$. Our first main result is a trichotomy: Every OMQ from $(\mathcal{EL}, \text{CQ})$ is in AC^0, NL-complete, or PTime-complete, and all three complexities actually occur already in $(\mathcal{EL}, \text{AQ})$. We consider this a remarkable sparseness of complexities. Let us illustrate the trichotomy using an example.

Example 3.1. *Consider an ontology that represents knowledge about genetic diseases, where* Disease1 *is caused by* Gene1 *and* Disease2 *by* Gene2. *A patient carries* Gene1 *if both parents carry* Gene1, *and the patient carries* Gene2 *if at least one parent carries* Gene2 *(dominant and recessive inheritance, respectively). Let $\mathcal{T}$ consist of the following concept inclusions:*

$$\begin{aligned}
\mathsf{Gene1} &\sqsubseteq \mathsf{Disease1},\\
\mathsf{Gene2} &\sqsubseteq \mathsf{Disease2}\\
\exists \mathsf{father}.\mathsf{Gene1} \sqcap \exists \mathsf{mother}.\mathsf{Gene1} &\sqsubseteq \mathsf{Gene1}\\
\exists \mathsf{father}.\mathsf{Gene2} &\sqsubseteq \mathsf{Gene2}\\
\exists \mathsf{mother}.\mathsf{Gene2} &\sqsubseteq \mathsf{Gene2}
\end{aligned}$$

For $\Sigma = \{\mathsf{Gene1}, \mathsf{Gene2}, \mathsf{mother}, \mathsf{father}\}$, the OMQ $Q_1 = (\mathcal{T}, \Sigma, \mathsf{Disease1}(x))$ is PTime-complete, while $Q_2 = (\mathcal{T}, \Sigma, \mathsf{Disease2}(x))$ is NL-complete. To see why it is harder to evaluate Q_1 compared to Q_2, note that for evaluating Q_1, one might have to verify the existence of a certain binary tree in the ABox whose leaves belong to Gene1, *while for evaluating Q_2, it is enough to verify the existence of a path to an individual belonging to* Gene2.

Our second main result is that for OMQs from $(\mathcal{EL}, \mathrm{CQ})$, evaluation in NL coincides with rewritability into linear Datalog. It is known that evaluation in AC^0 coincides with FO rewritability [Bie+16] and thus each of the three occurring complexities coincides with rewritability into a well-known database language: AC^0 corresponds to FO, NL to linear Datalog, and PTime to monadic Datalog. We also show that there is no constant bound on the arity of IDB relations in linear Datalog rewritings, that is, we find a sequence of OMQs from $(\mathcal{EL}, \mathrm{CQ})$ (and in fact, even from $(\mathcal{EL}, \mathrm{AQ})$) that are all rewritable into linear Datalog, but require higher and higher arities of IDB relations.

The second main result is proved using a characterization of linear Datalog rewritability in terms of bounded pathwidth that may be of independent interest. It is easiest to state for $(\mathcal{EL}, \mathrm{AQ})$: an OMQ Q is rewritable into linear Datalog (equivalently: can be evaluated in NL) if the class $\mathcal{M}_Q$ of the following ABoxes $\mathcal{A}$ has bounded pathwidth: $\mathcal{A}$ is tree-shaped, delivers the root as an answer to Q, and is minimal w.r.t. set inclusion regarding the latter property. For $(\mathcal{EL}, \mathrm{CQ})$, we have to replace in $\mathcal{M}_Q$ tree-shaped ABoxes with pseudo tree-shaped ones in which the root is an ABox that can have any relational structure, but whose size is bounded by the size of the actual query in q. These results are closely related to results on bounded pathwidth obstructions of CSPs, see for example [Dal05; DK08; CDK10].

Finally, we consider the meta problems associated to the studied properties of OMQs, such as whether a given OMQ is rewritable into linear Datalog, NL-hard, PTime-hard, etc. Each of these problems turns out to be ExpTime-complete, both in $(\mathcal{EL}, \mathrm{CQ})$ and in $(\mathcal{EL}, \mathrm{AQ})$. In the case of linear Datalog rewritability, our results provide a way of constructing a concrete rewriting when it exists.

We introduce the chapter-specific preliminaries in Section 3.1 and then start with considering the OMQ language $(\mathcal{EL}, \mathrm{conCQ})$. In Section 3.2, we show that $(\mathcal{EL}, \mathrm{conCQ})$ enjoys a dichotomy between AC^0 and NL, using a notion of bounded depth that was introduced in [Bie+16]. In particular, it was shown in [Bie+16] that when the ABoxes in $\mathcal{M}_Q$ have bounded depth, then Q can be evaluated in AC^0. We prove that otherwise, we find certain gadget ABoxes (we say that Q *has the ability to simulate* REACH) that allow us to reduce the reachability problem in directed graphs, thus showing NL-hardness. In Section 3.3, we prove a dichotomy between NL and PTime, still for $(\mathcal{EL}, \mathrm{conCQ})$. We first show that if $\mathcal{M}_Q$ has unbounded pathwidth, then we can find certain gadget ABoxes (we say that Q *has the ability to simulate* PSA) that allow us to reduce the path accessibility problem, thus showing PTime-hardness. This result is similar to, but substantially more difficult than the NL-hardness result in Section 3.2. We then proceed by showing that if $\mathcal{M}_Q$ has bounded pathwidth, then we can construct a two-way alternating word automaton that accepts suitable representations of pairs $(\mathcal{A}, \mathbf{a})$ where $\mathcal{A}$ is an ABox of low pathwidth and $\mathbf{a}$ and answer to Q on $\mathcal{A}$. We further show how to convert this automaton into a linear Datalog rewriting, which yields NL complexity. Section 3.4 is concerned with extending both of our dichotomies to potentially disconnected CQs. In Section 3.5, we prove that there is an infinite family of OMQs that are linear Datalog rewritable but for which the width of IDB relations in linear Datalog rewritings is not bounded by a constant. This strengthens a result by [DK08] who establish an analogous statement for CSPs. In Section 3.6 we prove decidability and ExpTime-completeness of the meta problems. The upper bounds are established using the ability to simulate PSA from Section 3.3 and alternating tree automata.

This chapter is an extended version of [LS17b]. The main differences are that [LS17b] only treats atomic queries but no conjunctive queries, does not provide characterizations in terms of bounded pathwidth, and achieves less optimal bounds on the width of IDB relations in constructed linear Datalog programs.

3.1 Preliminaries

We introduce the pathwidth of ABoxes, some notions regarding CQs that homomorphically map into a tree, and a fundamental glueing construction for ABoxes.

Pathwidth

The pathwidth of a graph is a number that measures how similar the graph is to a path, where a lower pathwidth means the graph is more similar to a path. For instance, every path has pathwidth 1, and the complete graph with k nodes has pathwidth $k-1$. Several different measures from graph theory turn out to be equivalent or closely related to the pathwidth, namely the vertex separation number, the interval thickness and the node search number. Furthermore, pathwidth can be characterized in terms of certain games played on the graph. Details on these notions can be found in [Bod98].

The pathwidth of a graph can be defined using path decompositions. A *path decomposition* of a (directed or undirected) graph $G = (V, E)$ is a sequence $V_1, \dots, V_n$ of subsets of V, such that

- $V_i \cap V_k \subseteq V_j$ for $1 \leq i \leq j \leq k \leq n$ and
- $\bigcup_{i=1}^{n} V_i = V$.

A path decomposition $V_1, \dots, V_n$ is an (ℓ, k)*-path decomposition* if $\ell = \max\{|V_i \cap V_{i+1}| \mid 1 \leq i \leq n-1\}$ and $k = \max\{|V_i| \mid 1 \leq i \leq n\}$. The *pathwidth* of G, denoted $\mathsf{pw}(G)$, is the smallest integer k, such that G has a $(\ell, k+1)$-path decomposition for some ℓ. For an ABox $\mathcal{A}$, a sequence $V_1, \dots, V_n$ of subsets of $\mathsf{ind}(\mathcal{A})$ is a path decomposition of $\mathcal{A}$ if $V_1, \dots, V_n$ is a path decomposition of $G_{\mathcal{A}}$. We assign a pathwidth to $\mathcal{A}$ by setting $\mathsf{pw}(\mathcal{A}) := \mathsf{pw}(G_{\mathcal{A}})$.

An important result we are going to use is that the full binary tree of depth k has pathwidth $\lceil \frac{k}{2} \rceil$, so the pathwidth grows with the depth of the tree [Sch89].

Linear Datalog

Datalog is a rule-based query language that supports recursive queries. When we are interested in rewritability of OMQs, Datalog is an interesting target language for OMQs that are not FO rewritable. In this chapter, we also characterize rewritability of OMQs into a certain fragment of Datalog, called linear Datalog.

A *Datalog rule* ρ has the form $S(\mathbf{x}) \leftarrow R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)$, $n > 0$, where $S, R_1, \ldots, R_n$ are relational symbols of any arity and $\mathbf{x}, \mathbf{y}_i$ denote tuples of variables. We refer to $S(\mathbf{x})$ as the *head* of ρ and to $R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)$ as the *body*. Every variable that occurs in the head of a rule is required to also occur in its body. Thus, syntactically, a datalog rule is the same as a CQ.

A *Datalog program* Π is a finite set of Datalog rules with a selected *goal relation* goal that does not occur in rule bodies.

The *arity of* Π, denoted ar(Π), is the arity of the goal relation. Relation symbols that occur in the head of at least one rule of Π are *intensional (IDB) relations*, and all remaining relation symbols in Π are *extensional (EDB) relations*. In our context, EDB relations must be unary or binary and are identified with concept names and role names. Note that, by definition, goal is an IDB relation. A Datalog program is *linear* if each rule body contains at most one IDB relation. The *width* of a Datalog program is the maximum arity of non-goal IDB relations used in it and its *diameter* is the maximum number of variables that occur in a rule in Π.

For an ABox $\mathcal{A}$ that uses no IDB relations from Π and a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathrm{ar}(\Pi)}$, we write $\mathcal{A} \models \Pi(a)$ if a is an answer to Π on $\mathcal{A}$, defined in the usual way [AHV95]: $\mathcal{A} \models \Pi(a)$ if goal(a) is a logical consequence of $\mathcal{A} \cup \Pi$ viewed as a set of first-order sentences (all variables in rules quantified universally). We also admit body atoms of the form $\top(x)$ that are vacuously true. This is just syntactic sugar since any rule with body atom $\top(x)$ can equivalently be replaced by a set of rules obtained by replacing $\top(x)$ in all possible ways with an atom $R(x_1, \ldots, x_n)$ where R is an EDB relation and where $x_i = x$ for some i and all other x_i are fresh variables.

A Datalog program Π over EDB signature Σ is a *rewriting* of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ if for all Σ-ABoxes $\mathcal{A}$ and all $\mathbf{a} \in \mathsf{ind}(\mathcal{A})$, $\mathcal{A} \models Q(\mathbf{a})$ if and only if $\mathcal{A} \models \Pi(\mathbf{a})$. We say that Q is *(linear) Datalog rewritable* if there is a (linear) Datalog program that is a rewriting of Q. It is well-known that all OMQs from $(\mathcal{ELI}, \mathrm{CQ})$ are Datalog rewritable.

Similarly to the derivations introduced in Section 2.8, we introduce derivations for Datalog, which are tree-shaped structures that encode a proof for a fact of the form goal($\mathbf{a}$).

Let Π be a Datalog program, $\mathcal{A}$ an ABox and $\mathbf{a}$ a tuple from $\mathsf{ind}(\mathcal{A})$. A *derivation of* $\Pi(\mathbf{a})$ *in* $\mathcal{A}$ is a labelled directed tree (V, ℓ) where

1. $\ell(x_0) = \mathrm{goal}(\mathbf{a})$ for x_0 the root node;
2. for each $x \in V$ with children $y_1, \ldots, y_k$, $k > 0$, there is a rule $S(\mathbf{y}) \leftarrow p(\mathbf{x})$ in Π and a substitution σ of variables by individuals from $\mathcal{A}$ such that $\ell(x) = S(\sigma\mathbf{y})$ and $\ell(y_1), \ldots, \ell(y_k)$ are exactly the facts in $p(\sigma\mathbf{x})$;
3. if x is a leaf, then $\ell(x) \in \mathcal{A}$.

It is well known that $\mathcal{A} \models \Pi(\mathbf{a})$ if and only if there is a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$.

We associate with each derivation $D = (V, \ell)$ of $\Pi(\mathbf{a})$ in $\mathcal{A}$ an ABox $\mathcal{A}_D$. In fact, we first associate an instance $\mathcal{A}_x$ with every $x \in V$ and then set $\mathcal{A}_D := \mathcal{A}_{x_0}$ for x_0 the root of D. If $x \in V$ is a leaf, then $\ell(x) \in \mathcal{A}$ and we set $\mathcal{A}_x = \{\ell(x)\}$. If $x \in V$ has children

$y_1, \dots, y_k$, $k > 0$, such that $y_1, \dots, y_\ell$ are non-leafs and $y_{\ell+1}, \dots, y_k$ are leafs, then $\mathcal{A}_x$ is obtained by starting with the assertions from $\ell(y_{\ell+1}), \dots, \ell(y_k)$ and then adding a copy of $\mathcal{A}_{y_i}$, for $1 \leq i \leq \ell$, in which all individuals except those in $\ell(x)$ are substituted with fresh individuals.

The following lemma is well known [AHV95] and easy to verify.

Lemma 3.2. *Let Π be a linear Datalog program and let D be a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$ and Π of diameter d. Then*

1. $\mathcal{A}_D \models \Pi(\mathbf{a})$*;*
2. *there is a homomorphism h from $\mathcal{A}_D$ to $\mathcal{A}$ with $h(\mathbf{a}) = \mathbf{a}$;*
3. *$\mathcal{A}_D$ has pathwidth at most d.*

Treeifying CQs.

A Boolean CQ q is *treeifiable* if there exists a homomorphism from q into a tree-shaped interpretation. With every treeifiable Boolean CQ q, we associate a tree-shaped CQ q^{tree} that is obtained by starting with q and then exhaustively *eliminating forks*, that is, identifying x_1 and x_2 whenever there are atoms $r(x_1, y)$ and $r(x_2, y)$. Informally, one should think of q^{tree} as the least constrained treeification of q. It is known that a CQ q is treeifiable if and only if the result of exhaustively eliminating forks is tree-shaped [Lut08]. Consequently, it can be decided in polynomial time whether a Boolean CQ is treeifiable. Figure 3.1 shows an example for a query that is treeifyable.

One reason for why treeification is useful is that every tree-shaped Boolean CQ q can be viewed as an $\mathcal{EL}$-concept C_q in a straightforward way. If, for example,

$$q() \leftarrow r(x, y) \wedge s(y, z) \wedge r(y, w) \wedge A(y) \wedge B(w),$$

then $C_q = \exists r.(A \sqcap \exists s.\top \sqcap \exists r.B)$.

A pair of variables (x, y) from a CQ q is *guarded* if q contains an atom of the form $r(x, y)$. For every guarded pair (x, y) and every $i \geq 0$, define $\text{reach}^i(x, y)$ to be the smallest set such that

1. $x \in \text{reach}^0(x, y)$ and $y \in \text{reach}^1(x, y)$;
2. if $z \in \text{reach}^i(x, y)$, $i > 0$, and $r(z, u) \in q$, then $u \in \text{reach}^{i+1}(x, y)$;
3. if $u \in \text{reach}^{i+1}(x, y)$, $i > 0$ and $r(z, u) \in q$, then $z \in \text{reach}^i(x, y)$.

Moreover, $\text{reach}(x, y) = \bigcup_i \text{reach}^i(x, y)$. We use $\text{trees}(q)$ to denote the set of all (tree-shaped) CQs p^{tree} such that $p = q|_{\text{reach}(x,y)}$ for some guarded pair (x, y) with p treeifiable.

It is easy to verify that the number of CQs in $\text{trees}(q)$ is linear in $|q|$. We briefly argue that $\text{trees}(q)$ can be computed in polynomial time. The number of guarded pairs is linear in $|q|$. For each guarded pair (x, y), $\text{reach}(x, y)$ can clearly be computed in polynomial time. Moreover, exhaustively eliminating forks on $p = q|_{\text{reach}(x,y)}$ takes only polynomial time, which tells us whether p is treeifiable and constructs p^{tree} if this is the case.

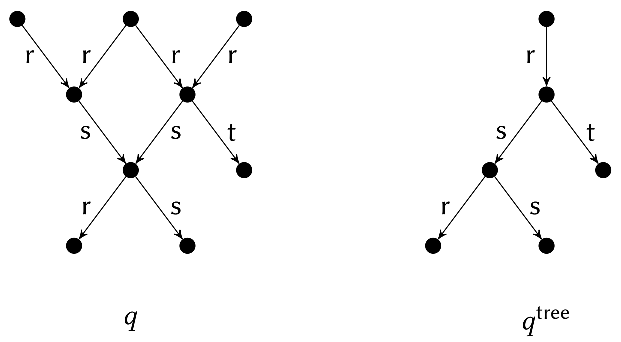

Figure 3.1: The left side shows a CQ q that is treeifyable. The right side shows q^{tree}, the unique CQ obtained from q by carrying out fork elimination.

Minimal Pseudo Tree-Shaped ABoxes

As mentioned in the introduction to this chapter, we shall often be interested in pseudo tree-shaped ABoxes $\mathcal{A}$ that give an answer $\mathbf{a}$ to an OMQ Q and that are minimal with this property regarding set inclusion, that is, no strict subset of $\mathcal{A}$ supports $\mathbf{a}$ as an answer to Q. We introduce some convenient notation for this. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{CQ})$. We use $\mathcal{M}_Q$ to denote the set of all pseudo tree-shaped Σ-ABoxes $\mathcal{A}$ of core size at most $|q|$ such that for some tuple $\mathbf{a}$ in the core of $\mathcal{A}$, $\mathcal{A} \models Q(\mathbf{a})$ while $\mathcal{A}' \not\models Q(\mathbf{a})$ for any $\mathcal{A}' \subsetneq \mathcal{A}$.

A Useful Lemma for Glueing ABoxes Together

We introduce a fundamental construction for merging ABoxes. Let $\mathcal{T}$ be an $\mathcal{ELI}$-TBox. For an ABox $\mathcal{A}$ and $a \in \mathsf{ind}(\mathcal{A})$, we use $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a)$ to denote the set of concept names A from $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models A(a)$, which is a $\mathcal{T}$-type. The following lemma allows us to glue together ABoxes under certain conditions.

Lemma 3.3. *Let $\mathcal{A}_1, \mathcal{A}_2$ be Σ-ABoxes and let $\mathcal{T}$ an $\mathcal{ELI}$-TBox such that $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}_1) \cap \mathsf{ind}(\mathcal{A}_2)$. Then $\mathsf{tp}_{\mathcal{A}_1 \cup \mathcal{A}_2,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}_i)$, $i \in \{1, 2\}$.*

Proof. Let $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{T}$ be as in the lemma. It suffices to show that $\mathsf{tp}_{\mathcal{A}_1 \cup \mathcal{A}_2,\mathcal{T}}(a) \subseteq \mathsf{tp}_{\mathcal{A}_i,\mathcal{T}}(a)$ for all $a \in \mathsf{ind}(\mathcal{A}_i)$, $i \in \{1, 2\}$. We show the contrapositive. Thus, assume that $\mathcal{A}_i, \mathcal{T} \not\models A(a)$ for some $i \in \{1, 2\}$. We have to show that $\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{T} \not\models A(a)$. Let $\mathcal{I}$ be the universal model of $\mathcal{T}$ and $\mathcal{A}_1 \cup \mathcal{A}_2$ and for each $j \in \{1, 2\}$, let $\mathcal{I}_j$ be the a universal model of $\mathcal{T}$ and $\mathcal{A}_j$. We can assume w.l.o.g. that $\Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathsf{ind}(\mathcal{A}_1) \cap \mathsf{ind}(\mathcal{A}_2)$. By assumption and since $\mathsf{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathsf{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$, we must have $a \notin A^{\mathcal{I}_1}$ and $a \notin A^{\mathcal{I}_2}$. Consider the (non-disjoint) union $\mathcal{I}$ of $\mathcal{I}_1$ and $\mathcal{I}_2$. Clearly, $\mathcal{I}$ is a model of $\mathcal{A}_1 \cup \mathcal{A}_2$ and $a \notin A^{\mathcal{I}}$. To show $\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{T} \not\models A(a)$, it thus remains to prove that $\mathcal{I}$ is a model of $\mathcal{T}$. To do this, we argue that all concept inclusions from $\mathcal{T}$ are satisfied:

- Consider $\exists r.A_1 \sqsubseteq A_2 \in \mathcal{T}$ and $a, b \in \Delta^{\mathcal{I}}$ such that $(a, b) \in r^{\mathcal{I}}$ and $b \in A_1^{\mathcal{I}}$. Then there exist $i, j \in \{1, 2\}$ such that $(a, b) \in r^{\mathcal{I}_i}$ and $b \in A_1^{\mathcal{I}_j}$. If $i = j$, then $a \in A_2^{\mathcal{I}}$, since $\mathcal{I}_i$ is a model of $\mathcal{T}$. Otherwise $b \in \Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathsf{ind}(\mathcal{A}_1) \cap \mathsf{ind}(\mathcal{A}_2)$, so by assumption,

$\mathrm{tp}_{\mathcal{A}_1,\mathcal{T}}(b) = \mathrm{tp}_{\mathcal{A}_2,\mathcal{T}}(b)$. It follows that $A_1 \in \mathrm{tp}_{\mathcal{A}_i,\mathcal{T}}(b)$ and thus, $b \in A_1^{\mathcal{I}_i}$. Together with $(a, b) \in r^{\mathcal{I}_i}$ and because $\mathcal{I}_i$ is a model of $\mathcal{T}$, it follows that $a \in A_2^{\mathcal{I}_i} \subseteq A_2^{\mathcal{I}}$. Thus, the inclusion $\exists r.A_1 \sqsubseteq A_2$ is satisfied in $\mathcal{I}$.

- Consider $\top \sqsubseteq A_1 \in \mathcal{T}$ and $a \in \Delta^{\mathcal{I}}$. Then $a \in \Delta^{\mathcal{I}_i}$ for some $i \in \{1, 2\}$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, we have $a \in A_1^{\mathcal{I}_i}$, so $a \in A_1^{\mathcal{I}}$ and the inclusion $\top \sqsubseteq A_1$ is satisfied in $\mathcal{I}$.
- Consider $A_1 \sqcap A_2 \sqsubseteq A_3 \in \mathcal{T}$ and $a \in A_1^{\mathcal{I}} \cap A_2^{\mathcal{I}}$. Then there are $i, j \in \{1, 2\}$ such that $a \in A_1^{\mathcal{I}_i}$ and $a \in A_2^{\mathcal{I}_j}$. If $i = j$, then $a \in A_3^{\mathcal{I}}$ follows, since $\mathcal{I}_i$ is a model of $\mathcal{T}$. Otherwise $a \in \Delta^{\mathcal{I}_1} \cap \Delta^{\mathcal{I}_2} = \mathrm{ind}(\mathcal{A}_1) \cap \mathrm{ind}(\mathcal{A}_2)$, so by assumption, $\mathrm{tp}_{\mathcal{A}_1,\mathcal{T}}(a) = \mathrm{tp}_{\mathcal{A}_2,\mathcal{T}}(a)$. For sure we have $A_1, A_2 \in \mathrm{tp}_{\mathcal{A}_1,\mathcal{T}}(a)$, so we have $a \in A_1^{\mathcal{I}_1} \cap A_2^{\mathcal{I}_1}$ and since $\mathcal{I}_1$ is a model of $\mathcal{T}$, we conclude $a \in A_3^{\mathcal{I}_1} \subseteq A_3^{\mathcal{I}}$, so the inclusion $A_1 \sqcap A_2 \sqsubseteq A_3$ is satisfied in $\mathcal{I}$.
- Consider $A_1 \sqsubseteq \exists r.A_2 \in \mathcal{T}$ and $a \in A_1^{\mathcal{I}}$. Then $a \in A_1^{\mathcal{I}_i}$ for some $i \in \{1, 2\}$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, we have $b \in \Delta^{\mathcal{I}_i}$ and $(a, b) \in r^{\mathcal{I}_i}$, hence also $b \in \Delta^{\mathcal{I}}$ and $(a, b) \in r^{\mathcal{I}}$ and thus, $A_1 \sqsubseteq \exists r.A_2$ is satisfied in $\mathcal{I}$.

□

3.2 AC^0 versus NL for Connected CQs

We prove a dichotomy between AC^0 and NL for $(\mathcal{EL}, \mathrm{conCQ})$ and show that for OMQs from this language, evaluation in AC^0 coincides with FO rewritability. The dichotomy does not depend on assumptions from complexity theory since it is known that AC$^0 \neq$ NL [FSS81]. We generalize the results obtained here to potentially disconnected CQs in Section 3.4.

FO rewritability of OMQs in $(\mathcal{EL}, \mathrm{CQ})$ has been characterized in [Bie+16] by a property called bounded depth. Informally, an OMQ Q has bounded depth if it looks only boundedly far into the ABox. To obtain our results, we show that unbounded depth implies NL-hardness. Formally, bounded depth is defined as follows. The *depth* of a tree-shaped ABox $\mathcal{A}$ is the largest number k such that there exists a directed path of length k starting from the root in $G_{\mathcal{A}}$. The *depth* of a pseudo tree-shaped ABox is the maximum depth of its trees. We say that an OMQ $Q \in (\mathcal{EL}, \mathrm{CQ})$ has *bounded depth* if there is a k such that every $\mathcal{A} \in \mathcal{M}_Q$ has depth at most k. If there is no such k, then Q has *unbounded depth.*

Theorem 3.4. *Let $Q \in (\mathcal{EL}, \mathrm{conCQ})$. The following are equivalent:*

(i) Q has bounded depth.

(ii) Q is FO-rewritable.

(iii) EVAL(Q) is in AC^0.

If these conditions do not hold, then EVAL(Q) is NL-hard under FO reductions.

The equivalence (ii) $\Leftrightarrow$ (iii) is closely related to a result in CSP. In fact, every OMQ of the form $(\mathcal{T}, \Sigma, \exists x A(x))$ with A a concept name and $\mathcal{T}$ formulated in $\mathcal{ELI}$ is equivalent to the complement of a CSP [Bie+14] and it is a known result in CSP that FO rewritability coincides with AC^0 [BKL08]. Conjunctive queries, however, go beyond the expressive power of (complements of) CSPs and thus we give a direct proof for (ii) $\Leftrightarrow$ (iii).

The equivalence (i) $\Leftrightarrow$ (ii) follows from Theorem 9 in [Bie+16]. Further, the implication (ii) $\Rightarrow$ (iii) is clear because first order formulas can be evaluated in AC^0. What remains to be shown is thus the implication (iii) $\Rightarrow$ (i) and the last sentence of the theorem. We show that unbounded depth implies NL-hardness, which establishes both since $AC^0 \neq$ NL.

We first give a rough sketch of how the reduction works. We reduce from REACH, the reachability problem in directed graphs, which is NL-complete under FO reductions [Imm99]. An input for this problem is a tuple $G = (V, E, s, t)$ where (V, E) is a directed graph, $s \in V$ a *source node* and $t \in V$ a *target node*. Such a tuple is a yes-instance if there exists a path from s to t in the graph (V, E). We further assume w.l.o.g. that $s \neq t$ and that the indegree of s and the outdegree of t are both 0, which simplifies the reduction.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ be an OMQ of unbounded depth. The reduction has to translate a tuple $G = (V, E, s, t)$ into a Σ-ABox $\mathcal{A}_G$ and a tuple $\mathbf{a}$ such that $\mathcal{A}_G \models Q(\mathbf{a})$ if and only if there is a path from s to t. We show that any ABox from $\mathcal{M}_Q$ of sufficiently large depth can be used to construct ABoxes $\mathcal{A}_{\text{source}}$, $\mathcal{A}_{\text{edge}}$ and $\mathcal{A}_{\text{target}}$ that can serve as gadgets in the reduction. More precisely, the ABox $\mathcal{A}_G$ has (among others) one individual a_v for every node $v \in V$, the edges of (V, E) will be represented using copies of $\mathcal{A}_{\text{edge}}$, and the source and target nodes will be marked using the ABoxes $\mathcal{A}_{\text{source}}$ and $\mathcal{A}_{\text{target}}$, respectively. We identify two $\mathcal{T}$-types t_0 and t_1 such that $\text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v) = t_1$ if v is reachable from s via a path in G and $\text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v) = t_0$ otherwise. The tuple $\mathbf{a}$ is then connected to a_t in a way such that $\mathcal{A}_G, \mathcal{T} \models q(\mathbf{a})$ if and only if $\text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_t) = t_1$.

We next define a property of Q, called the *ability to simulate* REACH, that makes the properties of $\mathcal{A}_{\text{source}}$, $\mathcal{A}_{\text{edge}}$, and $\mathcal{A}_{\text{target}}$ precise, as well as those of the $\mathcal{T}$-types t_0 and t_1. We then show that Q having unbounded depth implies the ability to simulate REACH and that this, in turn, implies NL-hardness via a reduction from REACH.

If M is a set of concept names, then $M(a)$ denotes the ABox $\{A(a) \mid A \in M\}$. We write $\mathcal{A}, \mathcal{T} \models M(a)$ to mean that $\mathcal{A}, \mathcal{T} \models A(a)$ for all $A \in M$. For every pseudo tree-shaped ABox $\mathcal{A}$ and a non-core individual $a \in \text{ind}(\mathcal{A})$, we use $\mathcal{A}^a$ to denote the tree-shaped ABox rooted at a. Note that every tree-shaped ABox is trivially pseudo tree-shaped with only one tree and where the core consists only of the root individual, so this notation can also be used if $\mathcal{A}$ is tree-shaped. Moreover, we use $\mathcal{A}_a$ to denote the pseudo tree-shaped ABox $\mathcal{A} \setminus \mathcal{A}^a$, that is, the ABox obtained from $\mathcal{A}$ by removing all assertions that involve descendants of a (making a a leaf) and all assertions of the form $A(a)$. We also combine these notations, writing for example $\mathcal{A}^a_{bc}$ for $((\mathcal{A}^a)_b)_c$.

Boolean queries require some special attention in the reduction since they can be made true by homomorphisms to anywhere in the universal model of $\mathcal{A}_G$ and $\mathcal{T}$, rather than to the neighborhood of the answer tuple $\mathbf{a}$ (recall that we work with connected CQs). We thus have to build $\mathcal{A}_G$ such that the universal model does not admit unintended homomorphisms. Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox of core size $|q|$ and $\mathbf{a}$ a tuple from $\text{ind}(\mathcal{A})$. We call a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ *core close* if there is some

$x \in \mathsf{var}(q)$ such that $h(x) \in \mathsf{ind}(\mathcal{A})$ is either in the core of $\mathcal{A}$ or $h(x)$ is an element in an anonymous tree rooted at a core individual. If $\mathsf{ar}(q) > 0$ and $\mathbf{a}$ is from the core of $\mathcal{A}$, then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close, but this is not true if q is Boolean.

Lemma 3.5. *Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$ be Boolean and $\mathcal{A} \in \mathcal{M}_Q$. Then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close.*

Proof. (sketch) Since $\mathcal{A} \in \mathcal{M}_Q$, $\mathcal{A}$ is minimal with the property that $\mathcal{A} \models Q$. Assume that there is a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is not core close. Since the anonymous trees in a universal model under an $\mathcal{EL}$ TBox are directed, there is no path in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ from any element in the range of h to any individual in the core of $\mathcal{A}$ (though a path in the converse direction might exist). Thus, we can remove all assertions in $\mathcal{A}$ that involve a core individual and the resulting ABox $\mathcal{A}'$ satisfies $\mathcal{A}' \models Q$, contradicting the minimality of $\mathcal{A}$. Formally, this can be proved by using Lemma 2.3 and showing that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ and $\mathcal{U}_{\mathcal{A}',\mathcal{T}}$ are isomorphic when restricted to non-core individuals and all elements reachable from them on a path. □

For the rest of this section, we assume w.l.o.g. that in any OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$, the TBox $\mathcal{T}$ has been modified as follows: for every $p \in \mathsf{trees}(q)$, introduce a fresh concept name A_p and add the concept inclusion $C_p \sqsubseteq A_p$ to $\mathcal{T}$ where C_p is p viewed as an $\mathcal{EL}$-concept. Finally, normalize $\mathcal{T}$ again. It is easy to see that the OMQ resulting from this modification is equivalent to the original OMQ Q. The extension is useful since $\mathcal{T}$-types become more informative, now potentially containing also the freshly introduced concept names. We are now ready to define the ability to simulate REACH.

Definition 3.6. *An OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$ has* the ability to simulate REACH *if there exist*

- *a pseudo tree-shaped Σ-ABox $\mathcal{A}$ of core size at most $|q|$,*
- *a tuple $\mathbf{a}$ from the core of $\mathcal{A}$ of length $\mathsf{ar}(q)$,*
- *a tree $\mathcal{A}_i$ of $\mathcal{A}$ with two distinguished non-core individuals b, c from $\mathsf{ind}(\mathcal{A}_i)$, where b has distance more than $|q|$ from the core, c is a descendant of b, and c has distance more than $|q|$ from b and*
- *$\mathcal{T}$-types $t_0 \subsetneq t_1$*

such that

1. *$\mathcal{A} \models Q(\mathbf{a})$,*
2. *$t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$,*
3. *$\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = t_0$,*
4. *$\mathcal{A}_b \cup t_0(b) \not\models Q(\mathbf{a})$ and*
5. *if q is Boolean, then every homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close.*

We define $\mathcal{A}_{\text{target}} = \mathcal{A}_b$, $\mathcal{A}_{\text{edge}} = \mathcal{A}_c^b$, and $\mathcal{A}_{\text{source}} = \mathcal{A}^c$.

To understand the essence of Definition 3.6, it is worthwhile to consider the special case where q is an AQ $A(x)$. In this case, Q has the ability to simulate REACH if there is a tree-shaped Σ-ABox $\mathcal{A}$ with root $a = \mathbf{a}$, two distinguished non-root individuals $b, c \in \text{ind}(\mathcal{A})$, c a descendant of b, and $\mathcal{T}$-types $t_0 \subsetneq t_1$ such that Conditions (1)-(4) of Definition 3.6 are satisfied. All remaining parts of Definition 3.6 should be thought of as technical complications induced by replacing AQs with CQs. The OMQ Q_2 from Example 3.1 has the ability to simulate REACH.

In Lemma 3.7 we show that unbounded depth implies the ability to simulate REACH and in Lemma 3.8 we show that the ability to simulate REACH enables a reduction from the reachability problem for directed graphs.

Lemma 3.7. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. If Q has unbounded depth, then Q has the ability to simulate REACH.*

Proof. We use a pumping argument. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have unbounded depth. There must be a pseudo tree-shaped ABox $\mathcal{A} \in \mathcal{M}_Q$ and a tuple $\mathbf{a}$ from its core such that $\mathcal{A} \models Q(\mathbf{a})$ and such that one of its trees, say $\mathcal{A}_i$, has depth at least $k := (|q| + 2) \cdot 3^{|\mathcal{T}|} + |q| + 2$. Consider a path of length at least k from the root of $\mathcal{A}_i$ to a leaf. Let $\mathcal{A}'$ denote the ABox obtained from $\mathcal{A}$ by removing all assertions that involve the leaf in this path. Since $\mathcal{A}$ is minimal, $\mathcal{A}' \not\models Q(\mathbf{a})$. Now, every individual b on the remaining path that has distance more than $|q|$ from the core is colored with the pair (t'_b, t_b) where $t'_b = \text{tp}_{\mathcal{A}',\mathcal{T}}(b)$ and $t_b = \text{tp}_{\mathcal{A},\mathcal{T}}(b)$. Observing $t'_b \subseteq t_b$, we obtain $3^{|\mathcal{T}|}$ as an upper bound for the number of different colors (t'_b, t_b) that may occur on the path. But the number of individuals on this path with distance more than $|q|$ from the core is $k - |q| - 1 = (|q| + 2) \cdot 3^{|\mathcal{T}|} + 1$, so by the pigeonhole principle there is one color (t', t) that appears $|q| + 2$ times on the path. Then there must be distinct individuals b and c that have distance more than $|q|$ from each other and such that $(t'_b, t_b) = (t'_c, t_c)$. W.l.o.g., let c be a descendant of b. We set $t_0 = t'_b$ and $t_1 = t_b$.

For this choice of $\mathcal{A}$, $\mathbf{a}$, b, c, t_0 and t_1, Conditions 1 and 2 from Definition 3.6 are immediately clear. With Lemma 3.3, we can replace $\mathcal{A}^c$ in $\mathcal{A}'$ by $t_0(c)$, so Condition 3 holds. Furthermore, we have $\mathcal{A}' \not\models Q(\mathbf{a})$ and $\text{tp}_{\mathcal{A}',\mathcal{T}}(b) = t_0$, so again by Lemma 3.3, if we replace $\mathcal{A}^b$ with $t_0(b)$, the types derived in the remaining ABox do not change, thus Condition 4 holds. Condition 5 follows from Lemma 3.5. □

Now for the reduction from REACH to EVAL(Q) when Q has the ability to simulate REACH.

Lemma 3.8. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. If Q has the ability to simulate REACH, then Q is NL-hard under FO reductions.*

Proof. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate REACH. Then there is a pseudo tree-shaped ABox $\mathcal{A}$, a tuple $\mathbf{a}$ in its core, distinguished individuals b and c, and types $t_0 \subsetneq t_1$ as in Definition 3.6. We reduce REACH to EVAL(Q). Let $G = (V, E, s, t)$ be an input tuple for REACH. We construct a Σ-ABox $\mathcal{A}_G$ that represents G. Reserve an individual a_v for every node $v \in V$. For every $(u, v) \in E$, include in $\mathcal{A}_G$ a copy $\mathcal{A}_{u,v}$ of $\mathcal{A}_{\text{edge}}$ that uses fresh individuals, identifying (the individual that corresponds to) c with a_u and b with a_v. Further include in $\mathcal{A}_G$ one copy of $\mathcal{A}_{\text{target}}$ that uses fresh individuals, identifying b with a_t, and one copy of $\mathcal{A}_{\text{source}}$ that uses fresh individuals, identifying c with a_s. W.l.o.g., we assume that the individuals in $\mathbf{a}$, added to $\mathcal{A}_G$ as part of the copy of $\mathcal{A}_{\text{target}}$, retain their original name. It can be verified that $\mathcal{A}_G$ can be constructed from G using an FO query, see [Imm99] for more information on FO reductions. It thus remains to show the following.

Claim 1. t is reachable from s in G if and only if $\mathcal{A}_G \models Q(\mathbf{a})$.

For the more straightforward "$\Rightarrow$" direction, let t be reachable from s. Then there is a path $s = v_0, \dots, v_n = t$ in G. By definition of $\mathcal{A}_G$, there is a copy of $\mathcal{A}_{\text{source}}$ whose root is a_s, so Condition 2 from Definition 3.6 yields $t_1 \subseteq \text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_s)$. Between any two $a_{v_i}, a_{v_{i+1}}$ there is a copy of $\mathcal{A}_{\text{edge}}$, so we inductively obtain $t_1 \subseteq \text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_{v_i})$ for all i. In particular, $t_1 \subseteq \text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_t)$. Finally, there is a copy of $\mathcal{A}_{\text{target}}$ in which b is identified with a_t. By Condition 1, we have $\mathcal{A}_G \models Q(\mathbf{a})$.

The "$\Leftarrow$" direction is more laborious. Assume that t is not reachable from s. Set

$$\begin{aligned} \mathcal{A}'_G \quad := \quad & \mathcal{A}_G \cup \{t_0(a_v) \mid v \in V \text{ is not reachable from } s\} \\ & \cup \{t_1(a_v) \mid v \in V \text{ is reachable from } s\}. \end{aligned}$$

We show that $\mathcal{A}'_G \not\models Q(\mathbf{a})$, which implies $\mathcal{A}_G \not\models Q(\mathbf{a})$.

We have defined $\mathcal{A}'_G$ as an extension of $\mathcal{A}_G$. Alternatively and more suitably for what we aim to prove, we can construct $\mathcal{A}'_G$ by starting with an ABox $\mathcal{A}_0$ that contains only the assertions $t_0(a_v)$ for all unreachable nodes $v \in V$ as well as $t_1(a_v)$ for all reachable nodes $v \in V$ and then exhaustively applying the following rules in an unspecified order, obtaining a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_m$ with $\mathcal{A}_m = \mathcal{A}'_G$:

1. Choose an edge $(u, v) \in E$ that has not been chosen before, take a copy $\mathcal{A}^{u,v}_{\text{edge}}$ of $\mathcal{A}_{\text{edge}}$ that uses fresh individuals names, with c renamed to a_u and b to a_v, and add the assertions $t_{\text{reach}(x)}(a_x)$ for $x \in \{u, v\}$ where $\text{reach}(x) = 1$ if x is reachable from s and $\text{reach}(x) = 0$ otherwise. Set $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{A}^{u,v}_{\text{edge}}$.

2. Introduce a copy $\mathcal{A}^s_{\text{source}}$ of $\mathcal{A}_{\text{source}}$ that uses fresh individual names, with b renamed to a_s, and add the assertions $t_1(a_s)$. Set $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{A}^s_{\text{source}}$.

3. Introduce a copy $\mathcal{A}^t_{\text{target}}$ of $\mathcal{A}_{\text{target}}$ that uses fresh individual names with b renamed to a_t, and add the assertions $t_0(a_t)$. Set $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{A}^t_{\text{target}}$.

Clearly, rule application terminates after $|E| + 2$ steps and results in the ABox $\mathcal{A}'_G$. Note that we add assertions $t_i(a)$, $i \in \{0, 1\}$ to the ABoxes constructed in the rules to enable application of the ABox glueing lemma, Lemma 3.3.

Claim 2. $\text{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_0$ if $u \in V$ is unreachable and $\text{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_1$ otherwise, for all $i \geq 0$.

The proof is by induction on i. For $i = 0$, the statement is clear. Now assume that the statement is true for some i and consider $\mathcal{A}_{i+1}$. If $\mathcal{A}_{i+1}$ was obtained by Rule 1, it follows from Conditions 2 and 3 of Definition 3.6 that $\mathsf{tp}_{\mathcal{A}^{u,v}_{\mathsf{edge}},\mathcal{T}}(a_x) = t_{\mathsf{reach}(x)}$ for all $x \in \{u, v\}$. So with Lemma 3.3 and since $\mathcal{A}_i$ and $\mathcal{A}^{u,v}_{\mathsf{edge}}$ share only the individuals a_u, a_v, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by Rule 2, we can use Condition 2 of Definition 3.6 and Lemma 3.3. In the case of Rule 3, it is clear that $\mathsf{tp}_{\mathcal{A}^t_{\mathsf{target}},\mathcal{T}}(a_t) = t_0$ and thus it remains to apply Lemma 3.3. This finishes the proof of Claim 2.

It remains to show that $\mathcal{A}'_G, \mathcal{T} \not\models q(\mathbf{a})$. Assume to the contrary that $\mathcal{A}'_G, \mathcal{T} \not\models q(\mathbf{a})$, that is, there is a homomorphism h from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ such that $h(\mathbf{x}) = \mathbf{a}$. There can be at most one individual of the form a_v in the range of h by construction of $\mathcal{A}'_G$ since b and c have distance exceeding $|q|$.

If there is no individual a_v in the range of h, then h only hits individuals from a single copy of $\mathcal{A}_{\mathsf{source}}$, $\mathcal{A}_{\mathsf{edge}}$, or $\mathcal{A}_{\mathsf{target}}$ as well as anonymous elements in the trees below them (since q is connected). First assume that this is $\mathcal{A}_{\mathsf{target}}$. By Claim 2 and since $\mathsf{reach}(t) = 0$, $\mathsf{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_t) = t_0$. It can be shown that the identity function is a homomorphism from $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}|_\Delta$, Δ the individuals from $\mathcal{A}_{\mathsf{target}}$ and anonymous elements below them, to $\mathcal{U}_{\mathcal{A}_{\mathsf{target}} \cup t_0(b),\mathcal{T}}$. By composing homomorphisms, it follows that $\mathcal{A}_{\mathsf{target}} \cup t_0(b), \mathcal{T} \models q(\mathbf{a})$, contradicting Condition 4.

Now assume that h only hits individuals from a copy of $\mathcal{A}_{\mathsf{source}}$ or $\mathcal{A}_{\mathsf{edge}}$ as well as anonymous elements in the trees below them. Then the restriction $\mathcal{U}$ of $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ to the range of h is tree-shaped. Moreover, q must be Boolean since the distance between $\mathbf{a}$ and the elements of $\mathcal{U}$ exceeds $|q|$ and q is treeifiable because h is a homomorphism to a tree-shaped interpretation. We have $d \in C_q^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ for the root d of $\mathcal{U}$. From Claim 2 and Lemma 3.3, it follows that some element of $\mathcal{U}_{\mathcal{A}_{\mathsf{source}} \cup t_1(c),\mathcal{T}}$ or of $\mathcal{U}_{\mathcal{A}_{\mathsf{edge}} \cup t_1(b) \cup t_1(c) \cup t_1(d),\mathcal{T}}$ also satisfies C_q. By Condition 2 and Lemma 3.3, the same is true for an element from $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is 'below' b (reachable from b by a directed path). Since the distance from the core to b in $\mathcal{A}$ exceeds $|q|$, this homomorphism is not core close, contradicting Condition 5.

Now assume that the range of h contains the individual a_v. Let $X_0 = \{x \in \mathsf{var}(q) \mid h(x) = a_v\}$ and let $X^\downarrow$ (resp. $X^\uparrow$) be the set of $x \in \mathsf{var}(q)$ such that $h(x)$ is some $a \in \mathsf{ind}(\mathcal{A}_G)$ or in an anonymous tree below such an a such that there exists a path of length at least one from a to a_v in $\mathcal{A}_G$ (resp. from a_v to a). We distinguish three cases.

Case 1: $v = t$, the target node. Since we assume that t has outdegree 0 in G, $h(x)$ is from the copy of $\mathcal{A}_{\mathsf{target}}$ or the attached anonymous trees for all $x \in X^\uparrow$ and $h(x)$ is from (potentially multiple) copies of $\mathcal{A}_{\mathsf{edge}}$ or the attached anonymous trees for all $x \in X^\downarrow$. It is thus possible to construct a homomorphism g from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ such that if $h(x) \in \mathsf{ind}(\mathcal{A}_G)$, then $g(x)$ is the individual $h(x)$ in $\mathcal{A}$ that $h(x)$ is a copy of. Then a_v being in the range of h implies that g is not core close.

Case 2: $v = s$, the source node. Since we assume that s has indegree 0 in G, $h(x)$ is from the copy of $\mathcal{A}_{\mathsf{source}}$ or the attached anonymous trees for all $x \in X^\downarrow$ and $h(x)$ is from (potentially multiple) copies of $\mathcal{A}_{\mathsf{edge}}$ or the attached anonymous trees for all $x \in X^\uparrow$. We can proceed as in the previous case.

Case 3: $v \notin \{s, t\}$. Then h hits (potentially multiple) copies of $\mathcal{A}_{\mathsf{edge}}$ and the attached anonymous trees. It is possible to construct a homomorphism g from $q|_{X^\uparrow \cup X_0}$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$

such that if $h(x) \in \mathsf{ind}(\mathcal{A}_G)$ and $x \in X^{\uparrow} \cup X_0$, then $g(x)$ is the individual in $\mathcal{A}$ that $h(x)$ is a copy of and, in particular, $g(x) = c$ for all $x \in X_0$. It remains to extend g to all of q. Let q' be obtained from q by identifying $x_1, x_2 \in \mathsf{var}(q)$ whenever $r(x_1, y), r(x_2, y) \in q$ and $x_1, x_2 \in X_0 \cup X^{\downarrow}$. It can be verified that the restriction of $\mathcal{A}_G$ to all elements between a_v and $\{h(x) \mid x \in X^{\downarrow}\}$ is a directed tree. Consequently, h is also a homomorphism from q' to $\mathcal{U}_{\mathcal{A}_G,\mathcal{T}}$. Moreover, $q' \setminus q|_{X^{\uparrow} \cup X_0}$ is the union of tree-shaped CQs $q_1, \dots, q_n$ that all share the same root x_0 and are otherwise variable disjoint. Each q_i can be viewed as an $\mathcal{EL}$-concept $\exists r.C$ such that $\exists r.C \sqsubseteq A_{\exists r.C}$ is in $\mathcal{T}$ and we must thus have $A_{\exists r.C} \in \mathsf{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v) \subseteq \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. Since $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c)$, we find a homomorphism from q_i to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that maps x_0 to c, for $1 \leq i \leq n$. Combining all these homomorphisms allows us to extend g to q', thus to q. □

This finishes the proof of Theorem 3.4.

3.3 NL versus PTime for Connected CQs

We prove a dichotomy between NL and PTime for $(\mathcal{EL}, \text{conCQ})$ and show that for OMQs from this language, evaluation in NL coincides with rewritability into linear Datalog. We also show that the latter two properties coincide with the OMQ having unbounded pathwidth, as defined below. We generalize our results to potentially disconnected CQs in Section 3.4.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$. We say that Q *has pathwidth at most* k if for every Σ-ABox $\mathcal{A}$ and tuple $\mathbf{a}$ with $\mathcal{A} \models Q(\mathbf{a})$, there is a Σ-ABox $\mathcal{A}'$ of pathwidth at most k such that $\mathcal{A}' \models Q(\mathbf{a})$ and a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Now Q has *bounded pathwidth* if it has pathwidth at most k for some k. If this is the case, we use $\mathsf{pw}(Q)$ to denote the smallest k such that Q has pathwidth at most k.

Theorem 3.9 (NL/PTime dichotomy). *Let $Q \in (\mathcal{EL}, \text{conCQ})$. The following are equivalent (assuming NL ≠ PTime):*

(i) Q has bounded pathwidth.

(ii) Q is rewritable into linear Datalog.

(iii) EVAL(Q) is in NL.

If these conditions do not hold, then EVAL(Q) is PTime-hard under FO reductions.

Remark 3.10. *Without the assumption NL ≠ PTime, Conditions (i) and (ii) are still equivalent to each other and they still imply (iii).*

The equivalence (i) ⇔ (ii) is closely related to a result in CSP. In fact, it is proved in [Dal05] that a CSP has an obstruction set of bounded pathwidth if and only if its complement is expressible in linear Datalog. From the viewpoint of the connection between OMQs and CSPs [Bie+14], obstructions correspond to homomorphic preimages of ABoxes and thus the result in [Dal05] implies (i) ⇔ (ii) for OMQs of the form $(\mathcal{T}, \Sigma, q)$, where q is a Boolean AQ of the form $q() \leftarrow A(x)$ and $\mathcal{T}$ formulated in $\mathcal{ELI}$. We give a direct proof of (i) $\Leftrightarrow$ (ii) in Section 3.3.2 to capture also CQs.

The implication (ii) $\Rightarrow$ (iii) is clear since every linear Datalog program can be evaluated in NL. It thus remains to prove the converse and the last sentence of the theorem. To achieve both and since we assume NL $\neq$ PTIME, it suffices to show that unbounded pathwidth implies PTIME-hardness. The structure of the proof is similar to the one for the dichotomy between AC^0 and NL in Section 3.2, but more sophisticated.

3.3.1 Unbounded Pathwidth Implies PTIME-hardness

We reduce from the well-known PTIME-complete problem *path systems accessibility* (PSA) [Imm99], closely related to alternating reachability on directed graphs and to the evaluation of Boolean circuits. An instance of PSA takes the form $G = (V, E, S, t)$ where V is a finite set of nodes, E is a ternary relation on V, $S \subseteq V$ is a set of *source nodes*, and $t \in V$ is a *target node*. A node $v \in V$ is *accessible* if $v \in S$ or there are accessible nodes u, w with $(u, w, v) \in E$. G is a yes-instance if the target node t is accessible. We assume w.l.o.g. that t does not appear in the first and second component of a triple in E, that no $s \in S$ appears in the third component of a triple in E, and that $t \notin S$.

The main difference to the NL-hardness proof in Section 3.2 is that instead of a gadget $\mathcal{A}_{\text{edge}}$ that transports a selected type t_1 from its input individual to its output individual, we now need a gadget $\mathcal{A}_\wedge$ with two input individuals and one output individual that behaves like a logical AND-gate. We formalize this as the ability to simulate PSA. Instead of proving directly that unbounded pathwidth of an OMQ Q implies that Q has the ability to simulate PSA, we first prove that unbounded pathwidth of Q implies unbounded branching of Q, that is, for any depth bound n, there is a pseudo tree-shaped ABox in $\mathcal{M}_Q$ that contains the full binary tree of depth n as a minor. In a second step, we then show that unbounded branching of Q implies that Q has the ability to simulate PSA. In fact, the ability to simulate PSA is actually equivalent to unbounded pathwidth and this is useful for the complexity analysis of the meta problems carried out in Section 3.6. We thus prove the converse directions as well. Finally, we show that the ability to simulate PSA implies PTIME-hardness.

We start with the formal definition of unbounded branching. Let $\mathcal{A}$ be a tree-shaped ABox. The *full binary tree of depth* k is the directed graph $G = (V, E)$ with $V = \{w \in \{1,2\}^* \mid 0 \leq |w| \leq k\}$ and $(v, w) \in E$ if $w = v1$ or $w = v2$. $\mathcal{A}$ *has the full binary tree of depth* k *as a minor* if there is a mapping f from the nodes of the full binary tree of depth k to $\mathsf{ind}(\mathcal{A})$ such that if $(v, w) \in E$, then $f(w)$ is a descendant of $f(v)$. We do usually not make the mapping f explicit but only say which individuals lie in the range of f. We are mostly interested in the largest k such that $\mathcal{A}$ has the full binary tree of depth k as a minor. This number, which we call the *branching number* of $\mathcal{A}$, denoted by $\mathsf{br}(\mathcal{A})$, can be easily computed by the following algorithm. Label every leaf of $\mathcal{A}$ with 0 and then inductively label the inner nodes as follows: If a is an inner node whose children have already been labeled and m is the maximum label of its children, label a with m if at most one child of a is labeled with m and label a with $m + 1$ if at least two children of a are labeled with m. It can be easily proved by induction on the co-depth of an individual that the label of a is equal to $\mathsf{br}(\mathcal{A}^a)$. In particular, $\mathsf{br}(\mathcal{A})$ is the label of the root of $\mathcal{A}$. We say that $Q \in (\mathcal{EL}, \mathrm{CQ})$ is *boundedly branching* if there exists a k such that for every pseudo tree-shaped ABox $\mathcal{A} \in \mathcal{M}_Q$ and every tree $\mathcal{A}_i$ in $\mathcal{A}$, we have $\mathsf{br}(\mathcal{A}_i) \leq k$. In that case, we define $\mathsf{br}(Q)$ to be the smallest such k. Otherwise, we call Q *unboundedly branching*.

Lemma 3.11. *Let $Q \in (\mathcal{EL}, \mathrm{CQ})$. Then Q has unbounded pathwidth if and only if Q is unboundedly branching.*

Proof. The "$\Leftarrow$" direction is clear since the full binary tree of depth k has pathwith $\lceil \frac{k}{2} \rceil$, see [Sch89]. For the "$\Rightarrow$" direction, we start by showing that for tree-shaped ABoxes, the branching number gives an upper bound on the pathwidth.

Claim. Let $\mathcal{A}$ be a tree-shaped ABox. Then there exists a (j, k)-path decomposition $V_1, \dots, V_n$ of $\mathcal{A}$ with $k \leq \mathsf{br}(\mathcal{A}) + 2$ and $j \leq k - 1$ such that the root of $\mathcal{A}$ is an element of V_n.

We prove the claim by induction on the depth of $\mathcal{A}$. If $\mathcal{A}$ has depth 0, then $\mathcal{A}$ has only one individual, $\mathsf{br}(\mathcal{A}) = 0$, and there is a trivial $(0, 1)$-path decomposition. If $\mathcal{A}$ has depth 1, then the root a of $\mathcal{A}$ has children $a_1, \dots, a_n$ with $n \geq 1$. We have $\mathsf{br}(\mathcal{A}) \leq 1$ and there is a $(1, 2)$-path decomposition $V_1, \dots, V_n$, where $V_i = \{a, a_i\}$.

If $\mathcal{A}$ has depth at least 2, let the root of $\mathcal{A}$ be called a and its children $a_1, \dots, a_m$. Let $V_1^i, \dots, V_{n_i}^i$ be the path decomposition of $\mathcal{A}^{a_i}$ that exists by induction hypothesis, for $1 \leq i \leq m$. We distinguish two cases:

- If $\mathsf{br}(\mathcal{A}) = \max\{\mathsf{br}(\mathcal{A}^{a_i}) \mid 1 \leq i \leq m\}$, then by definition of br, there is precisely one child a_i of a with $\mathsf{br}(\mathcal{A}^{a_i}) = \mathsf{br}(\mathcal{A})$. W.l.o.g. assume that $a_i = a_1$. Then $V_1^1, \dots, V_{n_1}^1, \{a, a_1\}, \{a\} \cup V_1^2, \dots, \{a\} \cup V_{n_2}^2, \{a\} \cup V_1^3, \dots, \{a\} \cup V_{n_3}^3, \dots, \{a\} \cup V_1^m, \dots, \{a\} \cup V_{n_m}^m$ is a path decomposition of $\mathcal{A}$ that fulfils the condition from the claim.
- If $\mathsf{br}(\mathcal{A}) = 1 + \max\{\mathsf{br}(\mathcal{A}^{a_i}) \mid 1 \leq i \leq m\}$, then $\{a\} \cup V_1^1, \dots, \{a\} \cup V_{n_1}^1, \{a\} \cup V_1^2, \dots, \{a\} \cup V_{n_2}^2, \dots, \{a\} \cup V_1^m, \dots, \{a\} \cup V_{n_m}^m$ is a path decomposition of $\mathcal{A}$ that fulfils the condition from the claim.

This finishes the proof of the claim.

We next show that for every OMQ $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{CQ})$, $\mathsf{br}(Q) = k$ implies $\mathsf{pw}(Q) \leq k + 2 + |q|$. Let Q be such an OMQ. Take a Σ-ABox $\mathcal{A}$ and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{A} \models Q(\mathbf{a})$. We have to show that there is a Σ-ABox $\mathcal{A}'$ of pathwidth at most k such that $\mathcal{A}' \models Q(\mathbf{a})$ and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. By Lemma 2.3, we obtain from $\mathcal{A}$ a pseudo tree-shaped Σ-ABox $\mathcal{A}'$ such that there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Clearly, $\mathcal{M}_Q$ contains a subset $\mathcal{A}''$ of $\mathcal{A}'$. We show that $\mathcal{A}''$ is as required, that is, the pathwidth of $\mathcal{A}''$ is at most k. From $\mathsf{br}(Q) = k$, $\mathcal{A}'' \models Q(\mathbf{a})$, and $\mathcal{A}'' \in \mathcal{M}_Q$, it follows that $\mathsf{br}(\mathcal{A}'') \leq k$. Let $\mathcal{A}'$ have core C and trees $\mathcal{A}_1, \dots, \mathcal{A}_m$. By the claim, every $\mathcal{A}_i$ has a $(j, k + 2)$-path decomposition $V_1^i, \dots, V_{n_i}^i$. Then we find a $(j + |q|, k + 2 + |q|)$-path decomposition of $\mathcal{A}'$: $\mathsf{ind}(C) \cup V_1^1, \dots, \mathsf{ind}(C) \cup V_{n_1}^1, \dots, \mathsf{ind}(C) \cup V_1^m, \dots, \mathsf{ind}(C) \cup V_{n_m}^m$. □

Our next goal is to identify suitable gadgets for the reduction from PSA. To achieve this, it is convenient to extend the TBox of the OMQ $Q = (\mathcal{T}, \Sigma, q)$ involved in the reduction. Recall that we have also used such an extension in the NL-hardness proof in Section 3.2 and that it has helped us to avoid unintended homomorphisms from the CQ to the (universal model of the) reduction ABox $\mathcal{A}_G$, in case the CQ is Boolean. Avoiding such homomorphisms is more complicated in the reduction of PSA which leads us to a different TBox extension that introduces $\mathcal{ELI}$-concepts. This is unproblematic since, as in Section 3.2, the OMQ based on the extended TBox is equivalent to the original one.

First assume that $\mathcal{T}$ is Boolean and treeifiable. A *role path between variables x and y in q* is a sequence of role names $r_1 \cdots r_n$ such that for distinct variables $x = x_0, \ldots, x_n = y$, q contains the atoms $r_1(x_1, x_2), \ldots, r_n(x_{n-1}, x_n)$. If q is treeifiable, then there are only polynomially many role paths in q^{tree}: the paths that occur in q^{tree}, the least constrained treeification of q defined in Section 3.1. Let C_q denote the set of $\mathcal{ELI}$-concepts of the form $\exists r_n^- . \cdots . \exists r_1^- . C$ where $r_1 \cdots r_n$ is a (potentially empty) role path in q^{tree} and C is $\top$ or a concept name from q or a CQ from $\text{trees}(q)$ viewed as an $\mathcal{EL}$-concept. Extend $\mathcal{T}$ with $C \sqsubseteq A_C$, A_C a fresh concept name, for all $C \in C_q$. Finally, normalize again. Clearly, the number of concept inclusions added to $\mathcal{T}$ is polynomial in $|q|$ and the resulting OMQ is equivalent to the original one.

Now assume that $\mathcal{T}$ is not Boolean and treeifiable. Then unintended homomorphisms are ruled out automatically. To prepare for the complexity analysis of the meta problems carried out in Section 3.6, however, we still carry out the same modification that we have also used in Section 3.2: For every $p \in \text{trees}(q)$, view p as an $\mathcal{EL}$-concept C and extend $\mathcal{T}$ with $C \sqsubseteq A_C$, A_C a fresh concept name.

The following lemma captures the use of the concepts C_q later on and gives an idea of why we use this particular set of concepts.

Lemma 3.12. *Let $q \in \text{conCQ}$ be Boolean and treeifiable, $\mathcal{I}_1, \mathcal{I}_2$ tree-shaped interpretations, and $d_i \in \Delta^{\mathcal{I}_i}$ for $i \in \{1, 2\}$ such that $d_1 \in C^{\mathcal{I}_1}$ implies $d_2 \in C^{\mathcal{I}_2}$ for all $C \in C_q$. If there is a homomorphism from q to $\mathcal{I}_1$ with d_1 in its range, then there is a homomorphism from q to $\mathcal{I}_2$ with d_2 in its range.*

Proof. Assume that there is a homomorphism h_1 from q to $\mathcal{I}_1$ with d_1 in its range. Since $\mathcal{I}_1$ is tree-shaped, this homomorphism factors into $h_1 = g_1 \circ h_q$, where h_q is the obvious homomorphism from q to q^{tree} and g_1 is a homomorphism from q^{tree} to $\mathcal{I}_1$. It clearly suffices to show that there is a homomorphism g_2 from q^{tree} to $\mathcal{I}_2$ with d_2 in its range.

Let $X_0 = g_1^{-1}(d_1)$. In a first step, we set $g_2(x) = d_2$ for all $x \in X_0$ and extend g_2 upwards as follows. Whenever $g_2(y)$ is already defined and there is an atom $r(x, y)$ in q^{tree}, define $g_2(x)$ to be the (unique) predecessor of $g_2(y)$ in $\mathcal{I}_2$. We show that g_2 is a homomorphism from $q^{\text{tree}}|_{\text{dom}(g_2)}$ to $\mathcal{I}_2$, $\text{dom}(g_2)$ the domain of g_2. If $r(x, y) \in q^{\text{tree}}$ with $g_2(x), g_2(y)$ defined, then q^{tree} contains a role path $r_1 \cdots r_n$ from x_1 to $x_n \in X_0$ with $r_1 = r$. Thus, there is a concept $C = \exists r_n^- \ldots \exists r_1^- . \top \in C_q$ such that $d_1 \in C^{\mathcal{I}_1}$. It follows that $d_2 \in C^{\mathcal{I}_2}$ and since $\mathcal{I}_2$ is tree-shaped and by construction of g_2, this yields $(g_2(x), g_2(y)) \in r^{\mathcal{I}_2}$. The argument for atoms $A(x) \in q^{\text{tree}}$ where g_2 has been defined on x is similar, using concepts of the form $C = \exists r_n^- \ldots \exists r_1^- . A \in C_q$.

In a second step, we define g_2 on all the remaining variables. Whenever $g_2(z)$ is still undefined for some $z \in \text{var}(q^{\text{tree}})$, there must be some $r(x, y) \in q^{\text{tree}}$ such that $g_2(x)$ is already defined, $g_2(y)$ is not yet defined, and z is in $q^{\text{tree}}|_{\text{reach}(x,y)}$. Since q is connected, there must be a (potentially empty) role path $r_1 \cdots r_n$ in q^{tree} from x to a variable $x_0 \in X_0$. Thus, C_q contains $C = \exists r_n^-. \cdots \exists r_1^-.D \in C_q$ where D is the $\mathcal{EL}$ concept that corresponds to $q^{\mathsf{tree}}|_{\mathsf{reach}(x,y)}$ and since $g_1(x_0) = d_1$, we have $d_1 \in C^{\mathcal{I}_1}$. Consequently, $d_2 \in C^{\mathcal{I}_2}$. Since $\mathcal{I}_2$ is tree-shaped, this implies $g_2(x) \in D^{\mathcal{I}_2}$ and thus there is a homomorphism from $q|_{\mathsf{reach}(x,y)}$ to $\mathcal{I}_2$ that maps x to $g_2(x)$. We use this homomorphism to extend g_2 to all variables in $\mathsf{reach}(x, y)$. □

If $\mathcal{A}$ is a pseudo tree-shaped ABox and $b \in \mathsf{ind}(\mathcal{A})$ has distance at least n from the core, we define the *ancestor path of b up to length n* to be the unique sequence $r_1 r_2 \dots r_n$ of role names such that $r_1(b_1, b_2), r_2(b_2, b_3), \dots r_n(b_n, b) \in \mathcal{A}$.

Definition 3.13. *Let* $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{conCQ})$. *We say that* Q *has* the ability to simulate PSA *if there exist*

- *$\mathcal{T}$-types $t_0 \subsetneq t_1$,*
- *a pseudo tree-shaped Σ-ABox $\mathcal{A}$ of core size $|q|$,*
- *a tuple* $\mathbf{a}$ *from the core of $\mathcal{A}$ of length* $\mathsf{ar}(q)$,
- *a tree $\mathcal{A}_i$ in $\mathcal{A}$ with three distinguished non-core individuals b, c and d from* $\mathsf{ind}(\mathcal{A}_i)$ *where c and d are incomparable descendants of b and such that b has distance more than $|q|$ from the core and the individuals b, c and d have pairwise distance more than $|q|$ from each other*

such that

1. $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$;
2. $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(d)$;
3. $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models q(\mathbf{a})$;
4. $\mathsf{tp}_{\mathcal{A}_c \cup t_0(c),\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A}_d \cup t_0(d),\mathcal{T}}(b) = t_0$,
5. *if q is Boolean, then every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close and*
6. *if q is Boolean, then b, c and d have the same ancestor path up to length $|q|$.*

We define $\mathcal{A}_{\mathsf{target}} := \mathcal{A}_b$, $\mathcal{A}_\wedge := \mathcal{A}^b_{cd}$ *and* $\mathcal{A}_{\mathsf{source}} := \mathcal{A}^c$.

With c and d being 'incomparable' descendants of b, we mean that neither d is a descendant of c nor vice versa.

To understand the essence of Definition 3.13, it is worthwhile to consider the special case where q is an AQ with body $A(x)$. In this case, Q has the ability to simulate PSA if there is a tree-shaped Σ-ABox $\mathcal{A}$ with root $a = \mathbf{a}$, three distinguished non-root individuals $b, c, d \in \mathsf{ind}(\mathcal{A})$, c and d incomparable descendants of b, and $\mathcal{T}$-types $t_0 \subsetneq t_1$ such that Conditions (1)-(4) of Definition 3.13 are satisfied. Figure 3.2 shows an ABox witnessing the ability to simulate PSA for such an OMQ. All remaining parts of Definition 3.13 should be thought of as technical complications induced by replacing AQs with CQs. As a preliminary for showing that unbounded branching implies the ability to simulate PSA, we give the following combinatorial lemma.

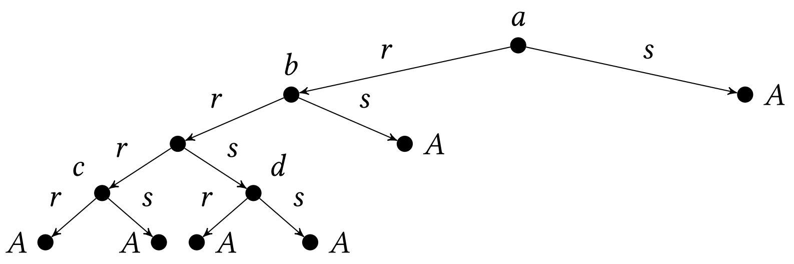

Figure 3.2: A witness ABox for the abilty to simulate PSA for the OMQ $Q = (\mathcal{T}, \Sigma, A(x))$, where $\mathcal{T} = \{\exists r.A \sqsubseteq B, \exists s.A \sqsubseteq C, B \sqcap C \sqsubseteq A\}$ and $\Sigma = \{r, s, A\}$.

Lemma 3.14. *Let T be a full binary tree of depth $n \cdot k \cdot d$ whose nodes are colored with n colors, $k \geq 0$ and $n, d \geq 1$. Then T has as a minor a monochromatic full binary tree of depth k such that any two distinct nodes of the minor have distance at least d from each other in T.*

Proof. Let T be a full binary tree of depth k whose nodes are colored with n colors. We assoicate T with a tuple $(m_1, \ldots, m_n)$ by letting, for $1 \leq i \leq m$, m_i be the minimum integer such that T does not have the color i monochromatic full binary tree of depth m_i as a minor. We prove the following.

Claim. $\sum_{i=1}^{n} m_i \geq k + 1$.

We proof the claim by induction on k. For $k = 0$, there is only one node, say of color i. Then clearly $\sum_{i=1}^{n} m_i = 1 \geq 1 = k + 1$.

Now assume that the claim holds for k and consider a tree T of depth $k + 1$, with associated tuple $(m_1, \ldots, m_n)$. Let a be the root of T and let the children of a root the subtrees T_1 and T_2, $(m^j_1, \ldots, m^j_n)$ the tuple associated with T_j for $j \in \{1, 2\}$. We distinguish two cases.

First assume that there exists a color j such that $m^1_j \neq m^2_j$. W.l.o.g. let $m^1_j < m^2_j$. Then $m_j = \max\{m^1_j, m^2_j\} > m^1_j$ and $m_i \geq m^1_i$ for all $i \neq j$. By the claim, $\sum_{i=1}^{n} m^1_i \geq k + 1$. It follows that $\sum_{i=1}^{n} m_i \geq k + 2$, as required.

Now assume that there is no such color j. Let i_0 be the color of a. From $m^1_{i_0} = m^2_{i_0}$, it follows that $m_{i_0} > m^1_{i_0}$ and thus we can proceed as before with i_0 in place of j. This finishes the proof of the claim.

The statement of the lemma now follows easily for $d = 1$: Let T be a full binary tree of depth $n \cdot k$ whose nodes are colored with n different colors. If there is no full monochromatic binary tree of depth k as a minor in T, then $m_i \leq k$ for all colors i, in contradiction to $\sum_{i=1}^{n} m_i \geq n \cdot k + 1$.

Now consider the case where $d > 1$. From the case $d = 1$, T contains as a minor a full monochromatic binary tree T' of depth $d \cdot k$. To obtain the desired full monochromatic binary tree T'' of depth k whose nodes have distance at least d from each other, we choose appropriate nodes from T'. Recall that the nodes of T' are $V = \{1, 2\}^k$. Then T'' can be constructed by choosing the nodes $V \cap \{1^d, 2^d\}^*$. Clearly, T'' is as required. □

Lemma 3.15. *Let $Q \in (\mathcal{EL}, \text{conCQ})$. Then Q has the ability to simulate* PSA *if and only if Q is unboundedly branching.*

Proof. "$\Rightarrow$". Assume that $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate PSA. Then there are $\mathcal{A}, \mathbf{a}, b, c, d, t_0$, and t_1 as in Definition 3.13. Let $k \geq 1$. We have to show that there is a pseudo tree-shaped Σ-ABox $\mathcal{A} \in \mathcal{M}_Q$ that has a tree $\mathcal{A}_i$ that has the full binary tree of depth k as a minor. We start with constructing an ABox $\mathcal{A}_0$ built up from the following set of ABoxes:

- one copy of $\mathcal{A}_{\text{target}}$;
- for every $w \in S$, one copy $\mathcal{A}_{\wedge,w}$ of $\mathcal{A}_\wedge$;
- for every $w \in \{0,1\}^k$, one copy $\mathcal{A}_{\text{source},w}$ of $\mathcal{A}_{\text{source}}$.

We identify the individual b of $\mathcal{A}_{\text{target}}$ with the individual b of $\mathcal{A}_{\wedge,\varepsilon}$. For every $w0 \in \{0,1\}^{k-1}$, we identify the individual b of $\mathcal{A}_{\wedge,w0}$ with the individual c of $\mathcal{A}_{\wedge,w}$ and for every $w1 \in \{0,1\}^{k-1}$, we identify the individual b of $\mathcal{A}_{\wedge,w1}$ with the individual d of $\mathcal{A}_{\wedge,w}$. Finally, for every $w0 \in \{0,1\}^k$, we identify the individual b of $\mathcal{A}_{\text{source},w0}$ with the individual c of $\mathcal{A}_{\wedge,w}$ and for every $w1 \in \{0,1\}^k$, we identify the individual b of $\mathcal{A}_{\text{source},w1}$ with the individual d of $\mathcal{A}_{\wedge,w}$. Since all $\mathcal{A}_\wedge$ and $\mathcal{A}_{\text{source}}$ are tree-shaped, the resulting ABox is $\mathcal{A}_0$ pseudo tree-shaped with the same core as $\mathcal{A}_{\text{target}}$.

It is clear that $\mathcal{A}_0$ has the full binary tree of depth k as a minor, formed by the set of roots of all $\mathcal{A}_{\wedge,w}$ and $\mathcal{A}_{\text{source},w}$. From Conditions 1 and 2 from Definition 3.13, it follows that $\mathcal{A}_0 \models Q(\mathbf{a})$. But there is no guarantee that $\mathcal{A}_0$ is minimal with this property, thus $\mathcal{A}_0$ need not be from $\mathcal{M}_Q$. Let $\mathcal{A}_0, \ldots \mathcal{A}_\ell$ be the sequence of ABoxes obtained by starting with $\mathcal{A}_0$ and exhaustively removing assertions such that $\mathcal{A}_i \models Q(\mathbf{a})$ still holds. We argue that the resulting ABox still has the full binary tree of depth k as a minor.

It suffices to show that role assertions connecting two individuals that lie on the same path from the core to a root of a $\mathcal{A}_{\text{source},w}$ are never removed. Assume to the contrary that such a role assertion is removed when transitioning from $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$. We distinguish three cases:

- The removed role assertion lies in $\mathcal{A}_{\wedge,w}$ on the path from b to c. Then $\text{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(b) \subseteq \text{tp}_{\mathcal{A}_c^b \cup t_0(c),\mathcal{T}}(b)$. By Condition 4 from Definition 3.13, the latter type is t_0. By iteratively using Conditions 3 and 4, it follows that $\text{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(b) = t_0$, with b the individual from the copy of $\mathcal{A}_{\text{target}}$. With Conditions 2 and 5, it follows that $\mathcal{A}_k \not\models Q(\mathbf{a})$. Contradiction.
- The removed role assertion lies in $\mathcal{A}_{\wedge,w}$ on the path from its b to its d. The proof is analogous.
- The removed role assertion lies in $\mathcal{A}_{\text{target}}$ on the path from the core to b. It follows that $\text{tp}_{\mathcal{A}_{i+1},\mathcal{T}}(a) \subseteq \text{tp}_{\mathcal{A}_b \cup t_0(b),\mathcal{T}}(a)$ for every individual a in the copy of $\mathcal{A}_{\text{target}}$. With Condition 3, it again follows that $\mathcal{A}_k \not\models Q(\mathbf{a})$.

"$\Leftarrow$". Assume that $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ is not boundedly branching. Let TP denote the set of all $\mathcal{T}$-types and set $m = 2^{|\mathcal{T}|}$. Clearly, $|\text{TP}| \leq m$. Set $k = m \cdot 2^m \cdot |\mathcal{T}|^{|q|} \cdot (2m^m + 1) \cdot |q|$. Since Q is not boundedly branching, we find a Σ-ABox $\mathcal{A} \in \mathcal{M}_Q$ and a tuple $\mathbf{a}$ from its core such that $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$ and one the trees $\mathcal{A}_i$ of $\mathcal{A}$ has the full binary tree of depth k as a minor. We show that $\mathcal{A}$ and $\mathbf{a}$ can serve as the ABox and tuple in Definition 3.13, that is, as a witness for Q having the ability to simulate PSA.

To identify the distinguished individuals b, c, d, we use a suitable coloring of the individuals of $\mathcal{A}_i$ and Lemma 3.14. In fact, we color every $b \in \mathsf{ind}(\mathcal{A}_i)$ with the color $(\mathsf{tp}_{\mathcal{A},\mathcal{T}}(b), S_b, r_1^b r_2^b \dots r_{|q|}^b)$ where $\mathsf{TP} \supseteq S_b = \{t \in S_b \mid \mathcal{A}_b \cup t(b), \mathcal{T} \models q(\mathbf{a})\}$ and where $r_1^b r_2^b \dots r_{|q|}^b$ is the ancestor path of b up to length $|q|$. There are no more than $m \cdot 2^m \cdot |\mathcal{T}|^{|q|}$ colors, so from Lemma 3.14 we know that $\mathcal{A}$ has as a minor a monochromatic full binary tree T of depth $2m^m + 1$ whose nodes have distance at least $|q|$ from each other. Let b be a child of the root of T (to make sure that b has depth at least $|q|$ from the core) and $T' \subseteq T$ the subtree of T rooted at b, so T' is a full binary tree of depth $2m^m$. We color every $c \in T'$ with the function $f_c : \mathsf{TP} \to \mathsf{TP}$ that is defined by $f_c(t) = \mathsf{tp}_{\mathcal{A}_c \cup t(c), \mathcal{T}}(b)$. There are at most m^m such functions, so again by Lemma 3.14, there will be the monochromatic binary tree of depth 2 as a minor. In particular, we find two incomparable individuals c and d in T' that are colored with the same function. We show that $\mathcal{A}$ we can find types t_1 and t_0 such that with the distinguished nodes b, c, d, $\mathcal{A}$ and $\mathbf{a}$ satisfy Conditions 1-6 from Definition 3.13.

Condition 1 is true by choice of $\mathcal{A}$. Set $t_1 := \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b)$. Then Condition 2 is satisfied because b, c and d were colored with the same color by the first coloring. For the same reason, Conditon 6 is fulfilled. Condition 5 follows from Lemma 3.5.

To define t_0, we first define a sequence $t'_0, t'_1, \dots$ of $\mathcal{T}$-types where $t'_0 = \emptyset$ and $t'_{i+1} = \mathsf{tp}_{\mathcal{A}_c \cup t'_i(c), \mathcal{T}}(b)$. It is clear that $t'_i \subseteq t'_{i+1}$ for all i. Let t_0 be the limit of the sequence. Since c and d were colored with the same function $f_c = f_d$, Condition 4 holds. It thus remains to argue that Condition 3 holds.

We show by induction on i that $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models q(\mathbf{a})$ for all $i \geq 0$. It is clear that $\mathcal{A}_c \cup t'_0(c), \mathcal{T} \not\models q(\mathbf{a})$ since $\mathcal{A}$ is minimal with $\mathcal{A}, \mathcal{T} \models q(\mathbf{a})$. Now assume that $\mathcal{A}_c \cup t'_i(c), \mathcal{T} \not\models q(\mathbf{a})$ for some i. Then $\mathcal{A}_c \cup t'_{i+1}(b), \mathcal{T} \not\models q(\mathbf{a})$. Since $S_b = S_c$ has been assured by the first coloring, we obtain $\mathcal{A}_c \cup t'_{i+1}(c), \mathcal{T} \not\models q(\mathbf{a})$ which completes the induction.

Thus $\mathcal{A}_c \cup t_0(c), \mathcal{T} \not\models q(\mathbf{a})$ and using again that $S_b = S_c$, we obtain Condition 3. □

It remains to show that the ability to simulate PSA implies PTime-hardness.

Lemma 3.16. *If $Q \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate* PSA*, then* EVAL*(Q) is PTime-hard under FO reductions.*

Proof. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{conCQ})$ have the ability to simulate PSA. Then there is a pseudo tree-shaped Σ-ABox $\mathcal{A}$, a tuple $\mathbf{a}$ in its core, distinguished individuals b, c and d, and types $t_0 \subsetneq t_1$ as in Definition 3.13. We reduce PSA to EVAL(Q).

Let $G = (V, E, S, t)$ be an input for PSA. We construct a Σ-ABox $\mathcal{A}_G$ that represents G. Reserve an individual a_v for every node $v \in V$. For every $(u, v, w) \in E$, include in $\mathcal{A}_G$ a copy $\mathcal{A}_{u,v,w}$ of $\mathcal{A}_\wedge$ that uses fresh individuals, identifying (the individual that corresponds to) c with a_u, d with a_v and b with a_w. For every $s \in S$ include in $\mathcal{A}_G$ one copy of $\mathcal{A}_{\text{source}}$ that uses fresh individuals, identifying c with a_s. Finally, include in $\mathcal{A}_G$ the ABox $\mathcal{A}_{\text{target}}$ identify b with a_t. (Note that we do not use a 'copy' of $\mathcal{A}_{\text{target}}$, so individuals from $\mathcal{A}_{\text{target}}$ except for b retain their name.) It can be verified that $\mathcal{A}_G$ can be constructed from G using an FO query. It thus remains to show the following.

Claim 1. t is accessible in G if and only if $\mathcal{A}_G \models Q(\mathbf{a})$.

For the "$\Rightarrow$" direction, assume that t is accessible in G. Define a sequence $S = S_0 \subseteq S_1 \subseteq \cdots \subseteq V$ by setting

$$S_{i+1} = S_i \cup \{w \in V \mid \text{there is a } (u, v, w) \in E \text{ such that } u, v \in S_i\}$$

and let the sequence stabilize at S_n. Clearly, the elements of S_n are exactly the accessible nodes. It can be shown by induction on i that whenever $v \in S_i$, then $t_1 \subseteq \text{tp}_{\mathcal{A}_G,\mathcal{T}}(a_v)$. In fact, the induction start follows from $t_1 = \text{tp}_{\mathcal{A},\mathcal{T}}(b)$ and the induction step from Condition 2 of Definition 3.13. It follows from Conditions 1 and 2 that $\mathcal{A}_{\text{target}} \cup t_1(b) \models Q(\mathbf{a})$, thus $\mathcal{A}_G \models Q(\mathbf{a})$ as required.

The "$\Leftarrow$" direction is more laborious. Assume that t is not accessible in G. Set

$$\begin{aligned}\mathcal{A}'_G \quad := \quad & \mathcal{A}_G \cup \{t_0(a_v) \mid v \in V \text{ is not accessible}\} \\ & \cup \{t_1(a_v) \mid v \in V \text{ is accessible}\}.\end{aligned}$$

We show that $\mathcal{A}'_G \not\models Q(\mathbf{a})$, which implies $\mathcal{A}_G \not\models Q(\mathbf{a})$.

We have defined $\mathcal{A}'_G$ as an extension of $\mathcal{A}_G$. Alternatively and more suitable for what we want to prove, $\mathcal{A}'_G$ can be obtained by starting with an ABox $\mathcal{A}_0$ that contains only the assertions $t_0(a_v)$ for all inaccessible nodes $v \in V$ as well as $t_1(a_v)$ for all accessible nodes $v \in V$, and then exhaustively applying the following rules in an unspecified order, obtaining a sequence of ABoxes $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_m$ with $\mathcal{A}_m = \mathcal{A}'_G$:

1. Choose a triple $(u, v, w) \in E$ that has not been chosen before, take a copy $\mathcal{A}_\wedge^{u,v,w}$ of $\mathcal{A}_\wedge$ using fresh individual names, with c renamed to a_u, d to a_v, and b to a_w, and add the assertions $t_{\text{acc}(x)}(a_x)$ for $x \in \{u, v, w\}$ where $\text{acc}(x) = 1$ if x is accessible and $\text{acc}(x) = 0$ otherwise. Set $\mathcal{A}i + 1 = \mathcal{A}_i \cup \mathcal{A}_\wedge^{u,v,w}$.

2. Choose a node $s \in S$ that has not been chosen before, introduce a copy $\mathcal{A}^s_{\text{source}}$ of $\mathcal{A}_{\text{source}}$ that uses fresh individuals, with c renamed to a_s, and add the assertions $t_1(a_s)$. Let the resulting ABox be called $\mathcal{A}^s_{\text{source}}$. Set $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{A}^s_{\text{source}}$.

3. Set $\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{A}'_{\text{target}}$, where $\mathcal{A}'_{\text{target}}$ is obtained from $\mathcal{A}_{\text{target}}$ by renaming b to a_t and adding the assertions $t_0(a_t)$.

Clearly, rule application terminates after finitely many steps and results in the ABox $\mathcal{A}'_G$. Note that we add assertions $t_i(a)$, $i \in \{0, 1\}$ to the ABoxes constructed in the rules to enable application of Lemma 3.3.

Claim 2. $\text{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_0$ if $u \in V$ is inaccessible and $\text{tp}_{\mathcal{A}_i,\mathcal{T}}(a_u) = t_1$ otherwise, for all $i \geq 0$.

The proof is by induction on i. For $i = 0$, the statement is clear since t_0 and t_1 are $\mathcal{T}$-types. Now assume the statement is true for some i and consider $\mathcal{A}_{i+1}$. If $\mathcal{A}_{i+1}$ was obtained by Rule 1, it can be verified using Conditions 2 and 4 from Definition 3.13 that $\text{tp}_{\mathcal{A}_\wedge^{u,v,w},\mathcal{T}}(a_x) = t_{\text{acc}(x)}$ for all $x \in \{u, v, w\}$. So with Lemma 3.3 and since $\mathcal{A}_i$ and $\mathcal{A}_\wedge^{u,v,w}$ share only the individuals u, v, w, the statement follows. If $\mathcal{A}_{i+1}$ was obtained by Rule 2, we can use Condition 2 and Lemma 3.3. If $\mathcal{A}_{i+1}$ was obtained by Rule 3, using $\text{acc}(t) = 0$

it can be verified that $\mathrm{tp}_{\mathcal{A}'_{\mathrm{target}},\mathcal{T}}(a_t) = t_0$ and with Lemma 3.3, the statement follows. This finishes the proof of the Claim 2.

It remains to show that $\mathcal{A}'_G, \mathcal{T} \not\models q(\mathbf{a})$. Assume to the contrary that $\mathcal{A}'_G, \mathcal{T} \models q(\mathbf{a})$, that is, there is a homomorphism h from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ such that $h(\mathbf{x}) = \mathbf{a}$. There can be at most one individual of the form a_v in the range of h by construction of $\mathcal{A}'_G$ and since b, c, d have distance more than $|q|$ from each other in $\mathcal{A}$.

If there is no individual a_v in the range of h, then h only hits individuals from a single copy of $\mathcal{A}_{\mathrm{source}}$, $\mathcal{A}_\wedge$, or $\mathcal{A}'_{\mathrm{target}}$ as well as anonymous elements in the trees below them (since q is connected). First assume that this is $\mathcal{A}'_{\mathrm{target}}$. By Claim 2 and since $\mathrm{acc}(t) = 0$, $\mathrm{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_t) = t_0$. It can be shown that the identity function is a homomorphism from $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}|_\Delta$, Δ the individuals from $\mathcal{A}'_{\mathrm{target}}$ and anonymous elements below them, to $\mathcal{U}_{\mathcal{A}_{\mathrm{target}} \cup t_0(b),\mathcal{T}}$. By composing homomorphisms, it follows that $\mathcal{A}_{\mathrm{target}} \cup t_0(b), \mathcal{T} \models q(\mathbf{a})$, contradicting Condition 3.

Now, assume that h only hits individuals from a copy of $\mathcal{A}_{\mathrm{source}}$ or $\mathcal{A}_\wedge$ as well as anonymous elements in the trees below them. Then the restriction $\mathcal{U}$ of $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ to the range of h is tree-shaped. Moreover, q must be Boolean since the distance between the core and the elements of $\mathcal{U}$ exceeds $|q|$ and q is treeifiable because h is a homomorphism to a tree-shaped interpretation. Since $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ is a model of $\mathcal{T}$, we have $d \in C_q^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ for the root d of $\mathcal{U}$, where C_q is q^{tree} seen as an $\mathcal{EL}$-concept. From Claim 2 and Lemma 3.3, it follows that some element of $\mathcal{U}_{\mathcal{A}_{\mathrm{source}} \cup t_1(c),\mathcal{T}}$ or of $\mathcal{U}_{\mathcal{A}_\wedge \cup t_1(b) \cup t_1(c) \cup t_1(d),\mathcal{T}}$ also satisfies C_q. By Condition 2 and Lemma 3.3, the same is true for an element from $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that is 'below' b (reachable from b by a directed path). Since the distance from the core to b in $\mathcal{A}$ exceeds $|q|$, this homomorphism is not core close, contradicting Condition 5.

Next, assume that the range of h contains a_v. Then q is Boolean since it is connected and the distance between the core in $\mathcal{A}'_{\mathrm{target}}$ and a_v exceeds $|q|$. Let $\mathcal{U}$ be the restriction of $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ to all elements within distance at most $|q|$ from a_v. Then $\mathcal{U}$ is almost tree-shaped except that a_v can have multiple predecessors. By Condition 6, however, there is a unique sequence of roles $r_n \cdots r_1$ such that in each path $d_m s_m \cdots d_2 s_2 d_1 s_1 a_v$ in $\mathcal{U}$, $s_m \cdots s_1$ is a postfix of $r_n \cdots r_1$. We can thus obtain a tree-shaped interpretation $\mathcal{U}'$ from $\mathcal{U}$ by exhaustively identifying elements d_1, d_2 whenever $(d_1, e), (d_2, e) \in r^{\mathcal{I}}$ for some e and r. Clearly, we can find a homomorphism h' from q to $\mathcal{U}'$. Consequently, q is treeifiable and the TBox has been extended with $C \sqsubseteq A_C$ for all $C \in \mathcal{C}_q$.

Claim 3. $a_v \in C^{\mathcal{U}'}$ if and only if $a_v \in C^{\mathcal{U}}$ for all $C \in \mathcal{C}_q$.

The "$\Leftarrow$" direction is immediate. For "$\Rightarrow$", assume to the contrary of what we aim to show that that $a_v \in C^{\mathcal{U}'}$ but $a_v \notin C^{\mathcal{U}}$ for some $C \in \mathcal{C}_q$. Then C has the form $\exists r_n^-.\cdots\exists r_1^-.C$ where $r_1 \cdots r_n$ is a (potentially empty) role path in q^{tree} and C is $\top$ or a concept name from q or a CQ from $\mathrm{trees}(q)$ viewed as an $\mathcal{EL}$-concept. In the former two cases, we clearly have $a_v \in C^{\mathcal{U}}$ by construction of $\mathcal{U}'$. In the latter case, C is of the form $\exists r.D$. Since $a_v \in C^{\mathcal{U}'}$, there is a path $d_1 r_1 d_2 \cdots r_{n-1} d_{n-1} r_n a_v$ in $\mathcal{U}'$ and $d_1 \in (\exists r.D)^{\mathcal{U}'}$. If there is an $e \neq d_2$ with $(d_1, e) \in r^{\mathcal{U}'}$ and $e \in D^{\mathcal{U}'}$, then again $a_v \in C^{\mathcal{U}}$ by construction of $\mathcal{U}'$. Assume that this is not the case, that is, $d_1 \in (\exists r.D)^{\mathcal{U}'}$ is true only because $r_1 = r$ and $d_2 \in D^{\mathcal{U}'}$. We may assume w.l.o.g. that C was chosen so that n is minimal, that is, there is no concept

$C' \in C_q$ with a shorter existential prefix than C such that $a_v \in C'^{\mathcal{U}'}$ but $a_v \notin C'^{\mathcal{U}}$. Let $D = A_1 \sqcap \cdots \sqcap A_{n_1} \sqcap \exists s_1.E_1 \cdots \sqcap \exists s_{n_2}.E_{n_2}$. Then $\exists r_n^-.\cdots\exists r_2^-.A_i$ and $\exists r_n^-.\cdots\exists r_2^-.\exists s_j.E_j$ are also in C_q for all relevant i and j. Let Γ be the set of all these concepts. We have $a_v \in G^{\mathcal{U}'}$ for all $G \in \Gamma$ and, since n is minimal, $a_v \in G^{\mathcal{U}}$ for all $G \in \Gamma$. By Claim 2, we have $\mathrm{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) \in \{t_0, t_1\}$. Let $\mathcal{B} = \mathcal{A}$ if $\mathrm{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) = t_1$ and $\mathcal{B} = \mathcal{A}_c$ if $\mathrm{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v) = t_0$. Then it follows from $\mathrm{tp}_{\mathcal{B},\mathcal{T}}(b) = \mathrm{tp}_{\mathcal{A}'_G,\mathcal{T}}(a_v)$ that $a_v \in A_C^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ if and only if $b \in A_C^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ for all $C \in C_q$. Since both $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ and $\mathcal{U}_{\mathcal{B},\mathcal{T}}$ are universal models and by construction of $\mathcal{T}$, $a_v \in C^{\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}}$ if and only if $b \in C^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ for all $C \in C_q$. By choice of $\mathcal{U}$, the same is true when $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ is replaced with $\mathcal{U}$. Thus, b satisfies all concepts from G as well as $\exists r_n^-.\cdots\exists r_1^-.\top$ in $\mathcal{U}_{\mathcal{B},\mathcal{T}}$. Since $\mathcal{U}_{\mathcal{B},\mathcal{T}}$ is tree-shaped, $b \in C^{\mathcal{U}_{\mathcal{B},\mathcal{T}}}$ and, consequently, $a_v \in C^{\mathcal{U}}$ as desired. This finishes the proof of Claim 3.

By Claims 2 and 3 and since both $\mathcal{U}_{\mathcal{A}'_G,\mathcal{T}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ are universal models and by construction of $\mathcal{T}$, $a_v \in C^{\mathcal{U}'}$ implies $b \in C^{\mathcal{U}_{\mathcal{A},\mathcal{T}}}$ for all $C \in C_q$. The same is true if we replace $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with its restriction $\mathcal{U}''$ to all elements that have distance at most $|q|$ from b. Note that $\mathcal{U}''$ is tree-shaped. We can apply Lemma 3.12 with $\mathcal{U}', a_v$ in place of $\mathcal{I}_1, d_1$ and $\mathcal{U}'', b$ in place of $\mathcal{I}_2, d_2$, obtaining a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with b in its range. Since the distance from the core to b in $\mathcal{A}$ exceeds $|q|$, this homomorphism is not core close, contradicting Condition 5. $\square$

3.3.2 Bounded Pathwidth Implies Linear Datalog Rewritability

We prove the equivalence (i) $\Leftrightarrow$ (ii) from Theorem 3.9. Our proof works even for OMQs from $(\mathcal{ELI}, \mathrm{CQ})$, that is, when inverse role are admitted in the TBox and when the conjunctive queries are not necessarily connected. We thus establish our result for this more general class of OMQs right away.

Lemma 3.17. *Let* $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELI}, \mathrm{CQ})$. *Then* Q *has bounded pathwidth if and only if* Q *is rewritable into linear Datalog. In the positive case, there exists a linear Datalog program of width* $\mathrm{pw}(Q) + \mathrm{ar}(q)$.

The "$\Leftarrow$" direction of Lemma 3.17 is easy to prove. Assume that $Q \in (\mathcal{ELI}, \mathrm{CQ})$ is rewritable into a linear Datalog program Π. We show that $\mathrm{pw}(Q)$ is at most the diameter d of Π. Take any pair $(\mathcal{A}, \mathbf{a})$ such that $\mathcal{A} \models Q(\mathbf{a})$. Since Π is a linear Datalog rewriting of Q, there exists a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}$. By Lemma 3.2, there exists an ABox $\mathcal{A}_D$ of pathwidth at most d such that $\mathcal{A}_D \models Q(\mathbf{a})$ and a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Hence, Q has pathwidth at most d.

The rest of this section takes care of the "$\Rightarrow$" direction of Lemma 3.17. Assume that Q has bounded pathwidth, say $\mathrm{pw}(Q) = k$. We obtain a linear Datalog program in the following way: We encode pairs $(\mathcal{A}, \mathbf{a})$ of an ABox $\mathcal{A}$ of pathwidth at most k and a tuple $\mathbf{a}$ from $\mathcal{A}$ as words over a finite alphabet, where one symbol of the word encodes one bag of the path decomposition of $\mathcal{A}$. We then construct an alternating two-way automaton on finite words that accepts precisely those words that encode a pair such that $\mathcal{A} \models Q(\mathbf{a})$. Such an automaton can always be transformed into a deterministic one-way automaton that accepts the same language [GO14]. From the latter automaton, we then construct the linear Datalog program that is equivalent to Q.

Two way alternating finite state automata. We introduce *two way alternating finite state automata (2AFAs).* For any set X, let $\mathcal{B}^+(X)$ denote the set of all positive Boolean formulas over X, i.e., formulas built using conjunction and disjunction over the elements of X used as propositional variables, and where the special formulas true and false are admitted as well. A 2AFA is a tuple $\mathfrak{A} = (S, \Gamma, \delta, s_0)$, where S is a finite set of *states*, Γ a finite alphabet, $\delta : S \times (\Gamma \cup \{\vdash, \dashv\} \to \mathcal{B}^+(\{\mathsf{left}, \mathsf{right}, \mathsf{stay}\} \times S)$ the *transition function* and $s_0 \in S$ the *initial state*. The two symbols $\vdash$ and $\dashv$ are used as the left end marker and right end marker, respectively, and it is required that $\delta(s, \vdash) \in \mathcal{B}^+(\{\mathsf{right}\} \times S)$ and $\delta(s, \dashv) \in \mathcal{B}^+(\{\mathsf{left}\} \times S)$ for all $s \in S$ so that the 2AFA can never leave the space of the input word.

For an input word $w = w_1 \dots w_n \in \Gamma^n$, define $w_0 = \vdash$ and $w_{n+1} = \dashv$. A *configuration* is a pair $(i, s) \in \{0, \dots, n+1\} \times S$. An *accepting run* of a 2AFA $\mathfrak{A} = (S, \Gamma, \delta, s_0)$ on w is a pair (T, r) that consists of a finite tree T and a labeling r that assigns a configuration to every node in T such that

1. $r(\varepsilon) = (1, s_0)$, where ε is the root of T and
2. if $m \in T$, $r(m) = (i, s)$, and $\delta(s, w_i) = \varphi$, then there is a (possibly empty) set $V \subseteq \{\mathsf{left}, \mathsf{right}, \mathsf{stay}\} \times S$ such that V (viewed as a propositional valuation) satisfies φ and for every $(\mathsf{left}, s') \in V$ there is a successor of m in T labeled with $(i-1, s')$, for every $(\mathsf{right}, s') \in V$ there is a successor of m in T labeled with $(i+1, s')$ and for every $(\mathsf{stay}, s') \in V$ there is a successor of m in T labeled with (i, s').

The *language accepted by a 2AFA* $\mathfrak{A}$, denoted by $L(\mathfrak{A})$, is the set of all words $w \in \Gamma^*$ such that there is an accepting run of $\mathfrak{A}$ on w. Note that there is no set of final states, acceptance is implicit via the transition function δ by using the formulas true and false. In particular, if there is a leaf labeled (i, s) in an accepting run, then $\delta(s, w_i) = \mathsf{true}$.

Construction of the 2AFA. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{ELI}, \mathrm{CQ})$ with $\mathrm{pw}(Q) = k$, and let $\mathbf{x} = x_1 \cdots x_{\mathsf{ar}(q)}$ be the answer variables in q. We encode pairs $(\mathcal{A}, \mathbf{a})$ with $\mathcal{A}$ a Σ-ABox of pathwidth at most k and $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ as words over a suitable finite alphabet Γ. Reserve a set $\mathsf{N} \subseteq \mathsf{N_I}$ of $2k+2$ individual names. Then Γ consists of all tuples $(\mathbf{b}, \mathcal{B}, \mathbf{c}, f)$, where

- $\mathcal{B}$ is a Σ-ABox with $|\mathsf{ind}(\mathcal{B})| \leq k$ that uses only individual names from N,
- $\mathbf{b}$ and $\mathbf{c}$ are tuples over $\mathsf{ind}(\mathcal{B})$ of arity at most k, and
- f is a partial function from $\mathbf{x}$ to $\mathsf{ind}(\mathcal{B})$.

Let $(\mathcal{A}, \mathbf{a})$ be a pair as described above with $\mathbf{a} = a_1 \cdots a_{\mathsf{ar}(q)}$ and let $V_1, \dots, V_n$ be a $(j, k+1)$ path decomposition of $\mathcal{A}$. We encode $(\mathcal{A}, \mathbf{a})$ by a word $(\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \cdots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$ from Γ^n, as follows:

- As $\mathcal{B}_1$, we use a copy of $\mathcal{A}|_{V_1}$ that uses only individual names from N.

- For $1 < i \leq n$, $\mathcal{B}_i$ is a copy of $\mathcal{A}|_{V_i}$ that uses the same individual names as $\mathcal{B}_{i-1}$ on $\mathsf{ind}(\mathcal{A}|_{V_{i-1}}) \cap \mathsf{ind}(\mathcal{A}|_{V_i})$ and otherwise only individual names from $\mathsf{N} \setminus \mathsf{ind}(\mathcal{B}_{i-1})$. Since bags have size at most $k + 1$, $|\mathsf{N}| = 2k + 2$ individual names suffice.
- $\mathbf{b}_1 = \mathbf{c}_n$ is the empty tuple.
- For $1 < i \leq n$, $\mathbf{b}_{i-1} = \mathbf{c}_i$ is the tuple that contains every individual from $\mathsf{ind}(\mathcal{B}_{i-1}) \cap \mathsf{ind}(\mathcal{B}_i)$ exactly once, ascending in some fixed order on N.
- For $1 \leq i \leq n$, f_i is defined as follows. If V_i contains a copy a_i' of a_i, then $f_i(x_i) = a_i'$; otherwise $f_i(x_i)$ is undefined.

It is easy to see that $(\mathbf{A}, \mathbf{a})$ can be recovered from w, and in particular $\mathbf{a}$ from f. Note that different words over Γ might encode the same pair $(\mathcal{A}, \mathbf{a})$, for example because we can choose different path decompositions, and there are words over Γ that do not properly encode a pair $(\mathcal{A}, \mathbf{a})$. Neither of this is problematic for the remaining proof.

We now construct a 2AFA $\mathfrak{A}$ that accepts a word that encode a pair $(\mathcal{A}, \mathbf{a})$ if and only if $\mathcal{A} \models Q(\mathbf{a})$. The idea is that an accepting run of the automaton has one main path on which it traverses the word from left to right, while guessing a homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$ in a stepwise fashion. The truth of all concept memberships in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that are necessary to realize this homomorphism is then checked by partial runs that branch off from the main path.

We now describe the set S of states of $\mathfrak{A}$. For the main path, we use states $s^g_{V,W}$ where $V \subseteq \mathsf{var}(q)$, $g : V \to \mathsf{N}$ is a partial function, and W is a subset of the binary atoms in q. Informally, the meaning of the state $s^g_{V,W}$ is that the variables from V have already been mapped to individuals in bags seen before, the binary atoms from W are already satisfied via this mapping, and g describes how variables are mapped to individuals that are in the intersection of the previous and the current bag. The initial state of $\mathfrak{A}$ is $s^g_{\emptyset,\emptyset}$ with g the empty map. We also use states s^a_A that make sure that the concept name A can be derived at $a \in \mathsf{N}$.

We have to take care of the fact that a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ can map existentially quantified variables to anonymous individuals, which are not explicitly represented in the input word. Let $\mathcal{B}$ be a Σ-ABox. A *partial q-match* in $\mathcal{B}$ is a partial function $h : \mathsf{var}(q) \to \mathsf{ind}(\mathcal{B}) \times \{\mathsf{named}, \mathsf{anon}\}$ such that if $x, y \in \mathsf{dom}(h)$, $r(x, y) \in q$ and $h_2(x) = h_2(y) = \mathsf{named}$, then $r(h_1(x), h_1(y)) \in \mathcal{B}$, where h_1 and h_2 are the projections of h to the first and second component, respectively. Informally, a partial q-match h partially describes a homomorphism g from q to $\mathcal{U}_{\mathcal{B},\mathcal{T}}$ where $h(x) = (a, \mathsf{named})$ means that $g(x) = a$ and $h(x) = (a, \mathsf{anon})$ means that $g(x)$ is some element in the subtree below a generated by the chase. Whether a part of the query can map into the anonymous part below some individual a only depends on the type realized at a. Define a relation $\mathcal{R} \subseteq \mathsf{TP} \times 2^{\mathsf{var}(q)} \times 2^{\mathsf{var}(q)}$ by putting $(t, V_1, V_2) \in \mathcal{R}$ if and only if $V_1 \subseteq V_2$, $V_2 \cap \mathbf{x} \subseteq V_1$ and there is a homomorphism from $q|_{V_2}$ to the universal model of the ABox $\{A(a) \mid A \in t\}$ and $\mathcal{T}$ that maps precisely the variables from V_1 to the root of the (tree-shaped) universal model.

An *explanation set* for a partial q-match $h : \mathsf{var}(q) \to \mathsf{ind}(\mathcal{B}) \times \{\mathsf{named}, \mathsf{anon}\}$ is a set Z of concept assertions that uses only individuals from $\mathsf{ind}(\mathcal{B})$ and satisfies the following conditions:

1. if $h(x) = (a, \text{named})$, then $\mathcal{B} \cup Z, \mathcal{T} \models A(h(x))$ for all $A(x) \in q$ and
2. if $h(x) = (a, \text{anon})$, then $(\{A \mid A(a) \in Z\}, h^{-1}(a, \text{named}), h_1^{-1}(a)) \in \mathcal{R}$.

Next, we describe the transition function δ. The following transitions are used for the main branch of automata runs:

$$\delta(s^g_{V,W}, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \bigvee_{h \in H} \left((\text{right}, s^{g_h}_{V_h,W_h}) \wedge \bigvee_{Z \in \mathcal{Z}_h} \left(\bigwedge_{A(a) \in Z} (\text{stay}, s^a_A) \right) \right)$$

where H is the set of all partial q-matches h for $\mathcal{B}$ such that $\text{dom}(g) \subseteq \text{dom}(h)$, h_1 and g agree on the intersection of their domains, and so do h_1 and f, and where

- g_h is h_1 restricted to answer variables x_i with $h_1(x_i)$ in $\mathbf{c}$,
- $V_h = V \cup \text{dom}(h)$,
- $V \cap \text{dom}(h) = \text{dom}(g)$,
- W_h is the union of W and all binary atoms from q that only use variables from h, and
- $\mathcal{Z}_h$ is the set of all explanation sets for h.

When the automaton reads the right end marker $\dashv$ and is in a state signifying that a complete homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ has been found, then we accept the input using the transition $\delta(s^g_{V,W}, \dashv) = \text{true}$ where $V = \text{var}(q)$, W is the set of all binary atoms of q, and g is the empty map.

The following transitions are used to verify the required concept memberships by checking for the existence of a suitable derivation tree (Lemma 2.4). Consider a state s^a_A and a symbol $(\mathbf{b}, \mathcal{B}, \mathbf{c}, f)$ such that $a \in \text{ind}(\mathcal{B})$. If $A(a) \in \mathcal{B}$, we set $\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \text{true}$. If a appears neither in $\mathbf{b}$ nor in $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = \bigvee_{\substack{Z \\ \mathcal{B} \cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b) \in Z} (\text{stay}, s^b_B)$$

If a appears in $\mathbf{b}$ but not in $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\text{left}, s^a_A) \vee \bigvee_{\substack{Z \\ \mathcal{B} \cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b) \in Z} (\text{stay}, s^b_B)$$

If a appears in $\mathbf{c}$ but not in $\mathbf{b}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\text{right}, s^a_A) \vee \bigvee_{\substack{Z \\ \mathcal{B} \cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b) \in Z} (\text{stay}, s^b_B)$$

If i appears in both $\mathbf{b}$ and $\mathbf{c}$:

$$\delta(s^a_A, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = (\text{left}, s^a_A) \vee (\text{right}, s^a_A) \vee \bigvee_{\substack{Z \\ \mathcal{B} \cup Z, \mathcal{T} \models A(a)}} \bigwedge_{B(b) \in Z} (\text{stay}, s^b_B)$$

Set $\delta(\cdot, \cdot) = \text{false}$ for all pairs from $S \times \Gamma$ that were not mentioned.

The automaton is now defined as $\mathfrak{A} = (S, \Gamma, \delta, s_0)$.

Lemma 3.18. *Let $\mathcal{A}$ be an ABox of pathwidth at most k, $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$, and $w \in \Gamma^*$ a word that encodes $(\mathcal{A}, \mathbf{a})$. Then $\mathcal{A} \models Q(\mathbf{a})$ if and only if $w \in L(\mathfrak{A})$.*

Proof. We start by proving that the states of the form s^a_A, used for checking the existence of a derivation for $A(a)$, work as in intended.

Claim. Let $a \in V_i$ and a' the name of its copy in the $\mathcal{B}_i$. Then there exists a successful run starting from the configuration $(s^{a'}_A, i)$ if and only if $\mathcal{A}, \mathcal{T} \models A(a)$.

First, assume that $\mathcal{A}, \mathcal{T} \models A(a)$. We have to construct a successful run starting from the configuration $(i, s^{a'}_A)$. By Lemma 2.4, there exists a derivation tree for $A(a)$. The statement can be proved by induction on the minimal number k such that $A(a)$ has a derivation tree of depth k. If $k = 0$, then $A(a) \in \mathcal{A}$, so $A(a') \in \mathcal{B}_i$, and in this case we have $\delta(s^a_A, (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)) = \mathsf{true}$, which means the run is successful. Now let $k > 0$ and consider a derivation tree for $A(a)$ of depth k.

- If the children of the root are of the form $B_1(a), \ldots, B_n(a)$ such that $\mathcal{T} \models B_1 \sqcap \ldots \sqcap B_n \sqsubseteq A$, then choose the set $Z = \{B_1(a'), \ldots, B_n(a')\}$ in the transition, so in the run, we add the children labeled with $(i, s^{a'}_{B_j})$ for all $1 \leq j \leq n$. By induction hypothesis, from all these configurations there exists a successful run, so these runs can be combined to obtain a successful run for $(i, s^{a'}_A)$.
- If the root has one child labeled $B(b)$ and we have $\mathcal{T} \models \exists r.B \sqsubseteq A$, then there exist $b \in \mathsf{ind}(\mathcal{A})$ and $r(a, b) \in \mathcal{A}$. This individual b does not necessarily lie in $\mathcal{B}_i$, but by the properties of a path decomposition, there exists a bag V_j such that $a, b \in V_j$ and, since $a \in V_i$, we also have $a \in V_k$ for all k between i and j. We extend the run as follows: If $j < i$, then use the transition $(\mathsf{left}, s^{a'}_A)$ for $i - j$ times. If $j > i$, then use the transition $(\mathsf{right}, s^{a'}_A)$ for $j - i$ times. Then we are in the configuration $(j, s^{a'}_A)$ and if we choose $Z = \{B(b')\}$, we can extend the run successfully by the induction hypothesis.

For the other direction, assume that there is a successful run starting from the configuration $(i, s^{a'}_A)$. We have to argue that $\mathcal{A}, \mathcal{T} \models A(a)$. The proof is by induction on the depth of the run. If the run has depth 0, i.e. the configuration $(i, s^{a'}_A)$ does not have any successors, then we must have $\delta(s^{a'}_A, (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)) = \mathsf{true}$. This is only the case if $A(a') \in \mathcal{B}_i$, so $A(a) \in \mathcal{A}$ and clearly, $\mathcal{T}, \mathcal{A} \models A(a)$. Now assume the run has depth $k > 0$. If the root node has a successor labeled $(i - 1, s^{a'}_A)$ or $(i + 1, s^{a'}_A)$, by induction hypothesis we have $\mathcal{A}, \mathcal{T} \models A(a)$. If the root node does not have a successor of this kind, then there exists a set Z and successors $(i, s^{b'}_B)$ for all $B(b') \in Z$ such that $\mathcal{B} \cup Z \models A(a')$. By induction hypothesis, we have $\mathcal{A}, \mathcal{T} \models B(b)$ for all $B(b') \in Z$. Together, this gives $\mathcal{A}, \mathcal{T} \models A(a)$. This finishes the proof of the claim.

Now we are ready to prove the lemma.

"$\Rightarrow$". Let $\mathcal{A} \models Q(\mathbf{a})$ and let $(\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \ldots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$ be an encoding of $(\mathcal{A}, \mathbf{a})$ based on some $(j, k{+}1)$ path decomposition $V_1, \ldots, V_n$ of $\mathcal{A}$. There exists a homomorphism h_0 from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ that maps the answer variables to $\mathbf{a}$. We use h_0 to guide the accepting run of $\mathfrak{A}$ on the word encoding $(\mathcal{A}, \mathbf{a})$. In the i-th step of the main branch of the run, always choose the partial q-match $h \in H$ according to h_0, i.e. if $h_0(x) = a \in \mathsf{ind}(\mathcal{A}) \cap V_i$

then $h(x) = (a', \mathsf{named})$, and if $h_0(x) = b$ for some anonymous individual b that lies in the subtree below some $c \in \mathsf{ind}(\mathcal{A})$ then $h(x) = (c', \mathsf{anon})$. As the explanation set for h we can just choose $Z_h = \{A(a') \mid a \in V_i \text{ and } \mathcal{T}, \mathcal{A} \models A(a)\}$.

We argue that following these choices, the main path will be successful, i.e. the leaf of the main branch is labeled with $(s^g_{V,W}, \dashv)$ such that $V = \mathsf{var}(q)$, W is the set of all binary atoms of q and g the empty map. Let $x \in \mathsf{var}(q)$. Then either $h_0(x) \in \mathsf{ind}(\mathcal{A})$ or $h_0(x)$ is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$. If $h_0(x) \in \mathsf{ind}(\mathcal{A})$, then let V_i be the first bag such that $h_0(x) \in V_i$ and thus there is a copy of $h_0(x)$ in $\mathsf{ind}(\mathcal{B}_i)$. Thus, in the i-th step of the main branch, x is added to V. Similarly, if $h_0(x)$ is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$, then let V_i be the first bag such that $b \in V_i$. Again, one can conclude that x is added to V in the i-th step of the main branch. Overall, it follows that $V = \mathsf{var}(q)$. Now, let $r(x, y)$ be a binary atom from q. If both $h_0(x)$ and $h_0(y)$ are in $\mathsf{ind}(\mathcal{A})$, then, since $V_1, \dots, V_n$ is a path decomposition of $\mathcal{A}$, there exists a bag V_i such that $h_0(x), h_0(y) \in V_i$, so there exists a copy of $r(h_0(x), h_0(y))$ in $\mathcal{B}_i$ and in the i-th step of the main branch, $r(x, y)$ is added to W. If at least one of $h_0(x)$ and $h_0(y)$ is not in $\mathsf{ind}(\mathcal{A})$, but is an anonymous individual below some $b \in \mathsf{ind}(\mathcal{A})$, then either both $h_0(x)$ and $h_0(y)$ are mapped to anonymous individuals below b or one of them is mapped to b. In any case, $r(x, y)$ is added to W in the i-th step of the main branch. Overall, it follows that W is the set of all binary atoms of q. Finally, g must be the empty map, since $\mathbf{c}_n = \emptyset$.

It follows immediately from the claim above that the other paths will be successful as well, i.e. whenever $\mathcal{A}, \mathcal{T} \models A(a)$ for some $a \in V_i$, then there is a successful run that starts at $(s^{a'}_A, i)$. This concludes the proof of the first direction.

"$\Leftarrow$". Assume there is a successful run of $\mathfrak{A}$ on $w = (\mathbf{b}_1, \mathcal{B}_1, \mathbf{c}_1, f_1) \dots (\mathbf{b}_n, \mathcal{B}_n, \mathbf{c}_n, f_n)$. The run must have one main path with states of the form $s^g_{V,W}$. In every step of the main path, one partial q-match h together with an explanation set Z_h is chosen. From these partial q-matches we can construct a map h_0 from $\mathsf{var}(q)$ to the universal model of $\mathcal{A}$ and $\mathcal{T}$ in the following way: Whenever a partial q-match h maps a variable x to (a, named), we set $h_0(x) = a$. Whenever a partial q-match h maps a variable x to (a, anon), then consider the explanation set Z_h. By the definition of the explanation set, we have $(\{B \mid B(a) \in Z_h\}, h^{-1}(a, \mathsf{named}), h_1^{-1}(a)) \in \mathcal{R}$, so there exists a homomorphism from $h_1^{-1}(a)$ to the canonical model of $\{B \mid B(a) \in Z_h\}$ that maps precisely the variables from $h^{-1}(a, \mathsf{named})$ to the root, which is the homomorphism we use to build h_0. From the condition $V \cap \mathsf{dom}(h) = \mathsf{dom}(g)$ it follows that a partial q-match chosen later in the run will not assign a different image to a variable that has appeared earlier in the domain of a partial q-match, so h_0 is well defined.

We show that h_0 is indeed a homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $q(\mathbf{x}) = \mathbf{a}$. Since the main branch ends in a configuration $(s^g_{V,W}, \dashv)$, where $V = \mathsf{var}(q)$, we know that $\mathsf{dom}(h_0) = \mathsf{var}(q)$. We argue that every atom of q is satisfied by h_0.

- Let $A(x)$ be a unary atom from q such that $h_0(x) \in \mathsf{ind}(\mathcal{A})$. Let h be the partial q-match that determined $h_0(x)$, so $h(x) = (a', \mathsf{named})$ for some $a' \in \mathsf{ind}(\mathcal{B})$, and let Z_h be the explanation set chosen in the run, so the configuration has children labeled (i, s^b_B) for every $B(b) \in Z_h$. Since the run is successful and we know from the claim above that a partial run starting from the configuration $(i, s^{b'}_B)$ is successful if and only if $\mathcal{T}, \mathcal{A} \models B(b)$, we have $\mathcal{T}, \mathcal{A} \models B(b)$ for all $B(b') \in Z_h$ and thus, $\mathcal{T}, \mathcal{A} \models A(a)$.

- Let $r(x, y)$ be a binary atom from q such that both $h_0(x)$ and $h_0(y)$ are in $\mathsf{ind}(\mathcal{A})$. Since the main branch ends in the state $s^g_{V,W}$, where W is the set of all binary atoms from q, there must be one step in the main branch, where $r(x, y)$ has been added to W, say the i-th step. This means that both x and y lie in $\mathsf{dom}(h)$, where h is the partial q-match chosen in the i-th step. Since h respects role atoms, we have $r(h_0(x), h_0(y)) \in \mathcal{A}$.
- Let $A(x)$ be a unary atom from q such that $h_0(x)$ is an anonymous individual below some $a \in \mathsf{ind}(\mathcal{A})$. Let h be the partial q-match that determined $h_0(x)$, so $h(x) = (a', \mathsf{anon})$, and let Z_h be the explanation set chosen in the run. By definition of an explanation set, there is a partial homomorphism from q with x in its domain to the universal model of $\{B(a) \mid B(a) \in Z_h\}$, which was used to define h_0 on x, so we have $\mathcal{T}, \mathcal{A} \models A(h_0(x))$.
- Let $r(x, y)$ be a binary atom from q such that at least one of $h_0(x)$ and $h_0(y)$ is an anonymous individual. Then the argument is similar to the previous case.

□

Construction of the linear Datalog program. Since every 2AFA can be transformed into an equivalent deterministic finite automaton (DFA) [GO14], Lemma 3.18 also ensures the existence of a DFA $\mathfrak{A} = (Q, \Sigma, \delta, s_0, F)$ that a word that encodes a pair $(\mathcal{A}, \mathbf{a})$ if and only if $\mathcal{A} \models Q(\mathbf{a})$. We use $\mathfrak{A}$ to construct the desired linear Datalog rewriting of Q.

The idea for the program is to guess a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ up front and then verify that $\mathcal{A} \models Q(\mathbf{a})$ by simulating $\mathfrak{A}$. The program uses the states of $\mathfrak{A}$ as IDBs. Each of these IDBs can appear in any arity between $\mathsf{ar}(q)$ and $\mathsf{ar}(q) + k$ with k the pathwidth of Q—technically, this means that we have $k + 1$ different IDBs for every state, but we use the same symbol for all of them since the arity will always be clear from the context. The first $\mathsf{ar}(q)$ components of each IDB are used to store the tuple $\mathbf{a}$ while the other components are used to store the individuals that occur in both of two consecutive bags of a path decomposition of $\mathcal{A}$.

Start rules: Given the ABox $\mathcal{A}$, the program starts by guessing a tuple $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$ using the following rule:

$$s_0(x_1, \ldots, x_n) \leftarrow \top(x_1) \wedge \top(x_2) \wedge \cdots \wedge \top(x_n).$$

Transition rules: Consider any transition $\delta(s_1, (\mathbf{b}, \mathcal{B}, \mathbf{c}, f)) = s_2$. Let $\varphi_\mathcal{B}$ be $\mathcal{B}$ viewed as a conjunction of atoms with individual names viewed as variables and let $\mathbf{x}'$ be obtained from $\mathbf{x}$ by replacing every variable $x_i \in \mathsf{dom}(f)$ by the individual name $f(x_i) \in \mathsf{ind}(\mathcal{B})$, also here viewed as a variable. We then include the following rule:

$$s_2(\mathbf{x}', \mathbf{c}) \leftarrow s_1(\mathbf{x}', \mathbf{b}) \wedge \varphi_\mathcal{B}.$$

This rule says that if the DFA is in state s_1, the intersection between the last bag and the current bag is $\mathbf{b}$, and we see a homomorphic image of $\mathcal{B}$, then the DFA can transition into state s_2 and remember the tuple $\mathbf{c}$. Applying such a rule leaves the tuple $\mathbf{a}$ stored in the first $\mathsf{ar}(q)$ components unchanged, but some of the variables in $\mathbf{x}'$ can appear in $\varphi_{\mathcal{B}}$ to enforce that $\mathbf{a}$ is compatible with the mapping of the answer variables that is prescribed by f and used in the simulated run of $\mathfrak{A}$.

Goal rules: If $s \in F$, then include the following rule:

$$\mathsf{goal}(\mathbf{x}) \leftarrow s(\mathbf{x}).$$

Lemma 3.19. *Π is a rewriting of Q.*

Proof. Let $\mathcal{A}$ be a Σ-ABox and let $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^{\mathsf{ar}(q)}$. First assume that $\mathcal{A} \models Q(\mathbf{a})$. Since Q is of pathwidth k, there must be a Σ-ABox $\mathcal{A}'$ of pathwidth at most k such that $\mathcal{A}' \models Q(\mathbf{a})$ and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$. Let $w = w_1 \dots w_n \in \Gamma^*$ encode the pair $(\mathcal{A}', \mathbf{a})$ where $w_i = (\mathbf{b}_i, \mathcal{B}_i, \mathbf{c}_i, f_i)$, and assume that $w \in L(\mathfrak{A})$. There is an accepting run of $\mathfrak{A}$ on w and thus we find states $s_0, \dots, s_n$ of $\mathfrak{A}$ such that $\delta(s_i, w_i) = s_{i+1}$ for $0 \leq i < n$ and s_n is an accepting state. This yields a derivation of $\Pi(\mathbf{a})$ in $\mathcal{A}'$, as follows. First, use the start rule to derive $s_0(\mathbf{a})$. For the next n steps, use the rule introduced for the transitions $\delta(s_i, w_i) = s_{i+1}$. In this way, we derive $s_n(\mathbf{a})$ since the individuals in the first $\mathsf{ar}(q)$ components do not change when using the transition rules. Because $s_n \in F$, a goal rule can be applied to derive $\mathsf{goal}(\mathbf{a})$ and thus $\mathcal{A}' \models \Pi(\mathbf{a})$. It is well-known that answers to Datalog programs are preserved under ABox homomorphisms [AHV95] and there is a homomorphism from $\mathcal{A}'$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$, we obtain $\mathcal{A} \models \Pi(\mathbf{a})$ as desired.

For the converse direction, assume that $\mathcal{A} \models \Pi(\mathbf{a})$. Then there is a derivation D of $\Pi(\mathbf{a})$ in $\mathcal{A}$. Since Π is of diameter at most k, $\mathcal{A}_D$ has pathwidth at most k. Consider the encoding of $(\mathcal{A}_D, \mathbf{a})$ as a word $w \in \Gamma^*$, based on the path decomposition induced by D in the obvious way. By construction of Π, D must use a start rule for $\mathbf{a}$, then a number of transition rules, and then a goal rule. Using the way in which these rules are constructed, it can be verified that this yields an accepting run of $\mathfrak{A}$ on w. Thus $\mathcal{A}_D \models Q(\mathbf{a})$. It remains to recall that there is a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on $\mathbf{a}$, and that answers to OMQs from $(\mathcal{ELI}, \mathrm{CQ})$ are preserved under ABox homomorphisms [Bie+14]. □

3.4 The Trichotomy for Disconnected CQs

We now lift the trichotomy result that is provided by Theorems 3.4 and 3.9 from connected CQs to unrestricted CQs. To achieve this, we show that the complexity of an OMQ $Q = (\mathcal{T}, \Sigma, q)$ with q a disconnected CQ is precisely the complexity of the hardest OMQ $(\mathcal{T}, \Sigma, q')$ with q' a maximal connected component (MCC) of q, provided that we first have removed redundant MCCs from q.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \mathrm{CQ})$. We say that Q is *empty* if $\mathcal{A} \not\models Q(\mathbf{a})$ for all Σ-ABoxes $\mathcal{A}$ and tuples $\mathbf{a}$. Every empty OMQ is trivially FO rewritable. An MCC of q is *Boolean* if it contains no answer variables. We call Q *redundant* if there is a Boolean MCC of q such that the OMQ obtained from Q by dropping that MCC from q is equivalent to Q.

For proving the intended trichotomy result, it is clearly sufficient to consider OMQs that are non-empty and non-redundant.

Theorem 3.20. *Let $Q \in (\mathcal{EL}, \text{CQ})$. Then either*

1. *Q is FO rewritable and thus $\textsc{eval}(Q)$ is in AC^0 or*
2. *Q is not FO rewritable and $\textsc{eval}(Q)$ is NL-hard under FO reductions.*

Proof. Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$ be a non-redundant and non-empty OMQ and let $q_1(\mathbf{x}_1), \dots, q_n(\mathbf{x}_n)$ be the MCCs of $q(\mathbf{x})$.

If every OMQ $Q_i = (\mathcal{T}, \Sigma, q_i)$ is FO rewritable, then the conjunction of all these FO rewritings is an FO rewriting of Q. It thus suffices to show that otherwise, Q is NL-hard. Thus assume that some Q_i is not FO rewritable. Since q_i is connected, $\textsc{eval}(Q_i)$ is NL-hard under FO reductions by Theorem 3.4. We prove that $\textsc{eval}(Q)$ is NL-hard under FO reductions by giving an FO reduction from $\textsc{eval}(Q_i)$. Let $\mathcal{A}_i$ be a Σ-ABox and $\mathbf{a}_i$ a tuple from $\mathsf{ind}(\mathcal{A}_i)^{\mathsf{ar}(q_i)}$. Since Q is non-empty and non-redundant, for every $j \neq i$ we find a Σ-ABox $\mathcal{A}_j$ and a tuple $\mathbf{a}_j$ such that

1. $\mathcal{T}, \mathcal{A}_j \models q_j(\mathbf{a}_j)$ and
2. if q_i is Boolean, then $\mathcal{T}, \mathcal{A}_j \not\models q_i$.

Define $\mathcal{A}$ to be the disjoint union of $\mathcal{A}_1, \dots, \mathcal{A}_n$ and $\mathbf{a} = \mathbf{a}_1 \cdots \mathbf{a}_n$. Clearly, $\mathcal{A}$ and $\mathbf{a}$ can be defined by an FO query, so this is an FO reduction.

We have to show that $\mathcal{A}_i \models Q_i(\mathbf{a}_i)$ if and only if $\mathcal{A} \models Q(\mathbf{a})$. The "$\Rightarrow$" direction is clear by construction of $\mathcal{A}$ and $\mathbf{a}$. For "$\Leftarrow$", assume that $\mathcal{A} \models Q(\mathbf{a})$. This implies $\mathcal{A} \models Q_i(\mathbf{a}_i)$, so there is a homomorphism h from q_i to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}_i) = \mathbf{a}_i$. The universal model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is the disjoint union of the universal models $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}}$, $1 \leq j \leq n$. Since q_i is connected, the range of h lies completely inside one of the $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}}$. In fact, it must lie in $\mathcal{U}_{\mathcal{A}_i,\mathcal{T}}$. If q_i is not Boolean, this is the case because $h(\mathbf{x}_i) = \mathbf{a}_i$ is a tuple from $\mathcal{A}_i$. If q_i is Boolean, then this follows from $\mathcal{T}, \mathcal{A}_j \not\models q_i$ which implies $\mathcal{U}_{\mathcal{A}_j,\mathcal{T}} \not\models q_i$. We have thus shown that $\mathcal{U}_{\mathcal{A}_i,\mathcal{T}} \models q_i(\mathbf{a}_i)$, implying $\mathcal{A}_i \models Q_i(\mathbf{a}_i)$ by Lemma 2.1, as desired. We have shown that $(\mathcal{T}, \Sigma, q)$ is NL-hard. It follows that $(\mathcal{T}, \Sigma, q)$ is not FO rewritable [FSS81]. □

To lift the dichotomy between NL and PTime dichotomy including the equivalence of NL and linear Datalog rewritability, we first give a helpful lemma aboutlinear Datalog programs.

Lemma 3.21. *Let $\Pi_1, \dots, \Pi_n$ be linear Datalog programs. Then there exists a linear Datalog program Π of arity $\Sigma_{i=1}^n \mathsf{ar}(\Pi_i)$ such that for all ABoxes $\mathcal{A}$ and tuples $\mathbf{a}_1, \dots, \mathbf{a}_n$,*

$$\mathcal{A} \models \Pi_i(\mathbf{a}_i) \text{ for } 1 \leq i \leq n \text{ if and only if } \mathcal{A} \models \Pi(\mathbf{a}_1, \dots, \mathbf{a}_n) .$$

Proof. It suffices to give a proof for the case $n = 2$, the general case follows by repeatedly applying the lemma for $n = 2$. So let Π_1, Π_2 be linear Datalog programs. We assume w.l.o.g. that Π_1 and Π_2 use disjoint sets of variables. Define a program Π that contains the following rules:

- for all rule $S_i(\mathbf{x}_i) \leftarrow \varphi_i(\mathbf{x}_i, \mathbf{y}_i) \in \Pi_i$, $i \in \{1, 2\}$, such that neither φ_1 not φ_2 contains an EDB relation, the rule

$$(S_1, S_2)(\mathbf{x}_1, \mathbf{x}_2) \leftarrow \varphi_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \varphi_2(\mathbf{x}_2, \mathbf{y}_2);$$

- for each rule $S_i(\mathbf{x}_i) \leftarrow \varphi_i(\mathbf{x}_i, \mathbf{y}_i) \in \Pi_i$ and each IDB relation S_{3-i} from Π_{3-i}, $i \in \{1, 2\}$, the rule

$$(S_1, S_2)(\mathbf{x}_1, \mathbf{x}_2) \leftarrow \varphi_i(\mathbf{x}_i, \mathbf{y}_i) \wedge S_{3-i}(\mathbf{x}_{3-i})$$

  where $\mathbf{x}_{3-i}$ is a tuple of fresh variables;
- the rule

$$\mathsf{goal}(\mathbf{x}_1, \mathbf{x}_2) \leftarrow (\mathsf{goal}, \mathsf{goal})(\mathbf{x}_1, \mathbf{x}_2).$$

It can be verified that $\mathcal{A} \models \Pi(\mathbf{a}_1, \mathbf{a}_2)$ if and only if both $\mathcal{A} \models \Pi_1(\mathbf{a}_1)$ and $\mathcal{A} \models \Pi_2(\mathbf{a}_2)$, for all Σ-ABoxes $\mathcal{A}$ and tuples $\mathbf{a}_1$, $\mathbf{a}_2$. □

Theorem 3.22. *Let $Q \in (\mathcal{EL}, \text{CQ})$. The following are equivalent (assuming NL ≠ PTIME):*

1. *Q has bounded pathwidth;*
2. *Q is linear Datalog rewritable;*
3. *EVAL(Q) is in NL.*

If these conditions do not hold, then EVAL(Q) is PTIME-hard under FO reductions.

Proof. The equivalence of (1) and (2) has been shown in Lemma 3.17 and the implication (2) ⇒ (3) is clear. To finish the proof, we show that if (2) does not hold, then EVAL(Q) is PTIME-hard, proving the implication (3) ⇒ (2) as well as the last sentence of the theorem.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$ and assume that Q is not rewritable into linear Datalog. As before, we can assume Q to be non-empty non-redundant. Let $q_1(\mathbf{x}_1), \dots, q_n(\mathbf{x}_n)$ be the connected components of $q(\mathbf{x})$. By Theorem 3.9, every OMQ $Q_i = (\mathcal{T}, \Sigma, q_i)$ is either rewritable into linear Datalog or PTIME-hard. By Lemma 3.21, every $(\mathcal{T}, \Sigma, q_i)$ being rewritable into linear Datalog implies that also $(\mathcal{T}, \Sigma, q)$ is linear Datalog rewritable. Since this is not the case, there must be some Q_i that is not rewritable into linear Datalog and thus PTIME-hard.

Now we can show PTIME-hardness of EVAL(Q) by an FO reduction from EVAL(Q_i), exactly as in the proof of Theorem 3.20. □

3.5 Width Hierarchy for Linear Datalog Rewritability

The width of the linear Datalog rewritings constructed in Section 3.3 depends on pw(Q), so if Q has high pathwidth, then we end up with a linear Datalog rewriting of high width. We aim to show that this is unavoidable, that is, there is no constant bound on the width of linear Datalog rewritings of OMQs from $(\mathcal{EL}, \text{CQ})$ and in fact not even from $(\mathcal{EL}, \text{AQ})$. It is interesting to contrast this with the fact that every OMQ from $(\mathcal{EL}, \text{CQ})$

can be rewritten into a *monadic* (non-linear) Datalog program [Baa+17]. Our result strengthens a result by [DK08] who establish an analogous statement for CSPs. This does not imply our result: While every OMQ from $(\mathcal{EL}, \text{AQ})$ is equivalent to a CSP up to complementation [Bie+14], the converse is false and indeed the CSPs used by Dalmau and Krokhin are not equivalent to an OMQ from $(\mathcal{EL}, \text{AQ})$. Our main aim is to prove the following.

Theorem 3.23. *For every $k > 0$, there is an OMQ $Q_k \in (\mathcal{EL}, \text{AQ})$ that is rewritable into linear Datalog, but not into a linear Datalog program of width k.*

When constructing the OMQs $Q_1, Q_2, \ldots$ for Theorem 3.23, we would like the ABoxes in $\mathcal{M}_{Q_k}$ to contain *more and more branching*. Now that we only work with $(\mathcal{EL}, \text{AQ})$, $\mathcal{M}_{Q_k}$ consists of tree-shaped ABoxes rather than of pseudo tree-shaped ones. Intuitively, if the ABoxes from $\mathcal{M}_{Q_k}$ have a lot of branching, then a linear Datalog rewriting of Q_k needs large width to simultaneously collect information about many different branches. However, we want Q_k to be linear Datalog rewritable so by Lemma 3.11 the ABoxes from $\mathcal{M}_{Q_k}$ must still branch only boundedly. We thus construct Q_k such that $\text{br}(\mathcal{A}) \leq k$ for all $\mathcal{A} \in \mathcal{M}_{Q_k}$ while for every $n \geq 1$, $\mathcal{M}_{Q_k}$ contains an ABox $\mathcal{A}_k^n$ that takes the form of a tree of outdegree 2 and of depth n such that $\text{br}(\mathcal{A}_k^n) = k$ and $\mathcal{A}_k^n$ has the maximum number of leaves that any such ABox can have. To make the latter more precise, let $\ell_d^k(n)$ denote the maximum number of leaves in any tree that has degree d, depth n, and does not have the full binary tree of depth $k+1$ as a minor, $d, k, n \geq 0$. We then want $\mathcal{A}_k^n$ to have exactly $\ell_2^k(n)$ leaves, which ensures that it is maximally branching.

We now construct Q_k. For every $k \geq 1$, let $Q_k = (\mathcal{T}_k, \Sigma, A_k(x))$ where $\Sigma = \{r, s, t, u\}$ and

$$\begin{aligned}
\mathcal{T}_k \quad = \quad & \{\top \sqsubseteq A_0\} \cup \\
& \{\exists x.A_i \sqsubseteq B_{x,i} \mid x \in \{r,s,t,u\}, 0 \leq i \leq k-1\} \cup \\
& \{\exists x.B_{x,i} \sqsubseteq B_{x,i} \mid x \in \{r,s,t,u\}, 0 \leq i \leq k-1\} \cup \\
& \{B_{r,i} \sqcap B_{s,i} \sqsubseteq A_{i+1} \mid 0 \leq i \leq k-1\} \cup \\
& \{B_{t,i} \sqcap B_{u,i+1} \sqsubseteq A_{i+1} \mid 0 \leq i \leq k-1\}.
\end{aligned}$$

Each concept name A_i represents the existence of a full binary tree of depth i, that is, if A_i is derived at the root of a tree-shaped Σ-ABox $\mathcal{A}$, then $\mathcal{A}$ contains the full binary tree of depth i as a minor. The concept inclusions $\exists x.B_{x,i} \sqsubseteq B_{x,i}$ in $\mathcal{T}_k$ ensure that Q_k is closed under subdivisions of ABoxes, that is, if $\mathcal{A} \in \mathcal{M}_{Q_k}$ and $\mathcal{A}'$ is obtained from $\mathcal{A}$ by subdividing an edge into a path (using the same role name as the original edge), then $\mathcal{A} \models Q_k(a)$ if and only if $\mathcal{A}' \models Q_k(a)$ for all $a \in \text{ind}(\mathcal{A})$. Informally spoken, subdivision will allow us to assume that every (connected) rule body in a linear Datalog rewriting can only 'see' a single branching.

To provide a better understanding of the four role names used, we now explicitly define the ABoxes $\mathcal{A}_k^n$ mentioned above. We refer to non-leaf individuals by the combination of role names of their outgoing edges, e.g. an rs-individual is an individual that has one outgoing r-edge, one outgoing s-edge and no other outgoing edges. Let $n, k \geq 1$. If $n \leq k$, then $\mathcal{A}_k^n$ is the full binary tree of depth n, where every non-leaf is an rs-individual. If

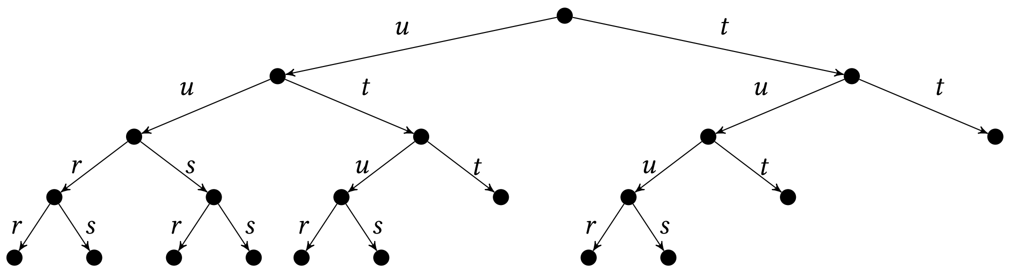

Figure 3.3: The ABox $\mathcal{A}_2^4$, which has depth 4, branching number 2 and lies in $\mathcal{M}_{Q_2}$. Since $4 > 2$, $\mathcal{A}_2^4$ is composed of one copy of $\mathcal{A}_2^3$ and one copy of $\mathcal{A}_1^3$ and a new tu-individual as root. This ABox has 11 leaves, which is the largest number of leaves that a binary tree of depth 4 can have, unless it contains the full binary tree of depth 3 as a minor.

$n > k = 1$, then $\mathcal{A}_k^n$ consists of a root that is a tu-individual where the t-successor is a leaf and the u-successor is the root of a copy of $\mathcal{A}_k^{n-1}$. Finally, for $n > k > 1$, take the disjoint union of $\mathcal{A}_{k-1}^{n-1}$ and $\mathcal{A}_k^{n-1}$ and introduce a new tu-individual as the root, the t-successor being the root of $\mathcal{A}_{k-1}^{n-1}$ and the u-successor the root of $\mathcal{A}_k^{n-1}$. As an example, Figure 3.3 shows $\mathcal{A}_2^4$.

Lemma 3.24. *For all $n, k \geq 1$,*

1. $\mathcal{A}_k^n \in \mathcal{M}_{Q_k}$;
2. $\mathsf{br}(\mathcal{A}_k^n) = k$;
3. *$\mathcal{A}_k^n$ has exactly $\ell_2^k(n)$ leaves.*

All three points can be proved by induction on n.

The following lemma establishes the first part of Theorem 3.23.

Lemma 3.25. *For every $k \geq 1$, Q_k is rewritable into linear Datalog.*

Proof. We show that $\mathsf{br}(Q_k) = k$, which implies rewritability into linear Datalog by Lemma 3.11 and Theorem 3.9.

Let $\mathcal{A} \in \mathcal{M}_{Q_k}$. We show that $\mathsf{br}(\mathcal{A}) = k$. First, let us analyse the types $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$, $a \in \mathsf{ind}(\mathcal{A})$, and the structure of $\mathcal{A}$. Since $\top \sqsubseteq A_0 \in \mathcal{T}_k$, none of the types $\mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ is empty. It is easy to verify that $\mathcal{T}_k \models A_i \sqsubseteq A_{i-1}$ and $\mathcal{T}_k \models B_{x,i} \sqsubseteq B_{x,i-1}$ for $1 \leq i \leq k$ and $x \in \{r, s, t, u\}$. We say that *a is of type i* if i is the largest integer such that $A_i \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$ and that *a is of x-type j_x* if j_x is the largest integer such that $B_{x,j_x} \in \mathsf{tp}_{\mathcal{A},\mathcal{T}_k}(a)$.

Claim 1. Every individual in $\mathcal{A}$ has degree at most two and every individual of degree two is an rs-individual or a tu-individual.

We first argue that every individual has at most one x-successor for every $x \in \{r, s, t, u\}$. Assume to the contrary that there exist three distinct individuals a, b, c and assertions $x(a, b), x(a, c) \in \mathcal{A}$ for some $x \in \{r, s, t, u\}$. Let b have type j and x-type ℓ and c have type m and x-type n. Then $B_{x,j}$, $B_{x,\ell}$, $B_{x,m}$ and $B_{x,n}$ are derived at a, but since $\mathcal{T}_k \models B_{x,i} \sqsubseteq B_{x,i-1}$

for $1 \leq i \leq k$, these four concept names are already implied by $B_{x,\max\{j,\ell,m,n\}}$. Thus one of the individuals b, c can be removed without altering the result of the query.

Now we argue that every individual with degree greater than one is either an rs-individual or a tu-individual. All other combinations do not appear due to the minimality of $\mathcal{A}$. For example, assume that there is an rst-node a. Then some $B_{r,j}, B_{s,\ell}, B_{t,m}$ are derived at a, assume that j, ℓ, m are maximal with this property. If a is the root of $\mathcal{A}$, then the t-edge can be removed. If a is not the root, it must be connected to its parent by a t-edge, since otherwise, the t-edge below a can be removed. So assume, a is a t-successor of its parent. If now $m \leq \mathsf{min}(j, \ell)$, the t-edge below a can be removed. If $m > \mathsf{min}(j, \ell)$, but then both the r-edge and the s-edge can be removed. In either case, $\mathcal{A}$ is not minimal. This finishes the proof of Claim 1.

Using minimality of $\mathcal{A}$, it can be argued that for every x-individual a (an indiviual with only one outgoing edge) with $x \in \{r, s, t, u\}$, there is some $b \in \mathsf{ind}(\mathcal{A})$ with $x(b, a) \in \mathcal{A}$ and it follows that a path from one branching point to the next is always a chain of the same role.

Claim 2. a is of type i if and only if $\mathsf{br}(\mathcal{A}^a) = i$ for all $a \in \mathsf{ind}(\mathcal{A})$ that are leaves or of degree two.

We prove the claim by induction on the number n of leaves $\mathcal{A}^a$. If $n = 1$, then a is a leaf itself, thus of type 0, and the statement follows. Now let $n > 1$ and let a be an individual of degree two with n leaves below it. We only argue the 'if' direction, the 'only if' direction can be argued similarly. So assume that $\mathsf{br}(\mathcal{A}^a) = i$ for some $i \geq 1$. Then by Claim 1, a has two outgoing paths that both reach two nodes b and c that are a leaf or of degree two. Let $j = \mathsf{br}(\mathcal{A}^b)$ and $\ell = \mathsf{br}(\mathcal{A}^c)$ and w.l.o.g. assume $j \geq \ell$. By induction hypothesis, b is of type j and c is of type ℓ. There are two possibilities: Either $j = \ell$, which implies $i = j + 1 = \ell + 1$, or $j > \ell$, which implies $i = j$. In case $j = \ell$, a must be an rs-individual. In fact, assuming a was a tu-individual, then $A_i(a)$ would be derived using $B_{t,i-1} \sqcap B_{u,i} \sqsubseteq A_i$, so a full binary tree of depth i below the t-successor of a is not needed and one could remove any leaf below the t-successor of a (contradicting minimality of $\mathcal{A}$), decreasing the depth of the largest binary tree minor by at most one. So since a is an rs-individual, $B_{r,i-1} \sqcap B_{s,i-1} \sqsubseteq A_i$ applies and a has type i. In case $j > \ell$, one can argue in a similar way that a must be a tu-individual and $j = \ell + 1$, and it follows that a has type i.

Since $\mathcal{A} \models Q_k(a)$ for the root a of $\mathcal{A}$, we know that a is of type k, so Claim 2 says that $\mathsf{br}(\mathcal{A}) = k$. □

The following proofs rely on an estimate of $\ell_d^k(n)$, namely on the fact that $\ell_d^k(n)$ as a function of n grows like a polynomial of degree k. This is established by the following lemma.

Lemma 3.26. $(d-1)^k(n-k)^k \leq \ell_d^k(n) \leq (k+1)(d-1)^k n^k$ *for all* $d, k \geq 0$ *and* $n \geq 2k$.

Proof. We aim to show that for all $d, k \geq 0$ and $n \geq 2k$,

$$\ell_d^k(n) = \sum_{i=0}^{k} (d-1)^i \binom{n}{i} \tag{$*$}$$

From (∗), the lower bound stated in the lemma is obtained by considering only the summand for $i = k$ and the upper bound is obtained by replacing every summand with the largest summand, which is the one for $i = k$ if $n \geq 2k$.

Towards proving (∗), we first observe that for all $n \geq 1$ and $k \geq 1$:

$$\ell_d^k(n) = \ell_d^k(n-1) + (d-1)\ell_d^{k-1}(n-1) \tag{$**$}$$

Let T be a tree with degree d and depth n that does not contain the full binary tree of depth $k+1$ as a minor and that has the largest possible number of leaves. It can easily be seen that the root of T has degree d and that T contains the full binary tree of depth k as a minor. Consider the subtrees $T_1, \ldots, T_d$ whose roots are the children of the root of T. There must be one of them that also has the full binary tree of depth k as a minor and all of them must have the full binary tree of depth $k-1$ as a minor, otherwise T would not have the maximum number of leaves. Moreover, there cannot be two subtrees that both have the full binary tree of depth k as a minor, since then T would have a minor of depth $k+1$. Since the number of leaves of T is the sum of the leaves of all T_j, (∗∗) follows.

Now we prove (∗) by induction on n. First observe that $\ell_d^k(0) = \ell_d^0(n) = 1$ for all d, k, n, thus (∗) holds for all cases where $k = 0$ or $n = 0$. Now let $k \geq 1$ and $n \geq 1$ and assume that (∗) holds for $\ell_d^k(n)$ and for $\ell_d^{k-1}(n)$. We show that it also holds for $\ell_d^k(n+1)$:

$$\begin{aligned}
\ell_d^k(n+1) &= \ell_d^k(n) + (d-1)\cdot\ell_d^{k-1}(n)\\
&= \sum_{i=0}^{k}(d-1)^i\binom{n}{i} + (d-1)\sum_{i=0}^{k-1}(d-1)^i\binom{n}{i}\\
&= \sum_{i=0}^{k}(d-1)^i\binom{n}{i} + \sum_{i=1}^{k}(d-1)^i\binom{n}{i-1}\\
&= 1 + \sum_{i=1}^{k}(d-1)^i\binom{n+1}{i}\\
&= \sum_{i=0}^{k}(d-1)^i\binom{n+1}{i}
\end{aligned}$$

□

Remark 3.27. *The numbers $\ell_2^k(n)$ are an interesting family of sequences that appear in several different settings in combinatorics [OEI]. The following table shows the values of $\ell_2^k(n)$ for small k and n.*

k \ n	0	1	2	3	4	5	6	7	8	9
0	1	1	1	1	1	1	1	1	1	1
1	1	2	3	4	5	6	7	8	9	10
2	1	2	4	7	11	16	22	29	37	46
3	1	2	4	8	15	26	42	64	93	130
4	1	2	4	8	16	31	57	99	163	256
5	1	2	4	8	16	32	63	120	219	382

Notice how every row starts with powers of two and, for values $n \le k$, pretends to be the function 2^n. This makes sense, since these are the number of leaves in a binary tree of depth n, where the depth of the forbidden minor is larger than n. After the k-th column, the forbidden minor comes into play, so the next tree is an almost full binary tree with only one leaf missing, resulting in a number of the form $2^n - 1$. In the end, the k-th row does not grow polynomially, but like a polynomial of degree k, as proven in Lemma 3.26.

To show that linear Datalog rewritings of the defined family of OMQs require unbounded width, we first show that they require unbounded diameter and then proceed by showing that the width of rewritings cannot be significantly smaller than the required diameter. To make the latter step work, we actually show the former on an infinite family of classes of ABoxes of restricted shape. More precisely, for all $i \ge 0$ we consider the class $\mathfrak{C}_i$ of all forest-shaped Σ-ABoxes in which the distance between any two branching individuals exceeds i (where a forest is a disjoint union of trees and a branching individual is one that has at least two successors). Since the queries Q_k are closed under subdivisions of ABoxes, each class $\mathfrak{C}_i$ contains ABoxes whose root is an answer to the query.

The idea for proving that any linear Datalog rewriting of Q_k requires high diameter is then as follows. We assume to the contrary that there is a linear Datalog rewriting Π of Q_k that has low diameter and consider the linear Datalog derivation of $Q_k(a)$ in some $\mathcal{A}_k^n$ with root a. A careful analysis shows that $\mathcal{A}_D$ contains a tree-shaped sub-ABox of depth n that has as many leaves as $\mathcal{A}_k^n$ and thus by Lemma 3.26 contains a deep full binary tree as a minor. Thus $\mathcal{A}_D$ has high pathwidth which contradicts the assumption that Π has low diameter.

Lemma 3.28. *For any $i \ge 0$, Q_{2k+3} is not rewritable into a linear Datalog program of diameter k on the class of ABoxes $\mathfrak{C}_i$.*

Proof. Let $i \ge 0$. For $n \ge k \ge 1$, denote by $\mathcal{B}_k^n$ the ABox obtained from $\mathcal{A}_k^n$ by subdividing every edge into a path of length $i+1$ of the same role. Note that $\mathcal{B}_k^n \in \mathfrak{C}_i$. Using Lemma 3.24, it is easy to see that $\mathcal{B}_k^n \in \mathcal{M}_{Q_k}$ and $\mathcal{B}_k^n$ has $\ell_2^k(n)$ leaves, so from Lemma 3.26 it follows that $\mathcal{B}_k^n$ has at least $(n-k)^k$ leaves.

For the sake of contradiction, assume that there is a $k \ge 1$, such that Q_{2k+3} is rewritable into a linear Datalog program Π of diameter k on the class $\mathfrak{C}_i$. Choose n very large (we will make this precise later) and let $\mathcal{A} = \mathcal{B}_{2k+3}^n$, so $\mathcal{A} \in \mathcal{M}_{Q_{2k+3}}$, it has depth $n(i+1)$ and at least $(n-2k-3)^{2k+3}$ leaves.

Let a_0 be the root of $\mathcal{A}$. We have $\mathcal{A}, \mathcal{T} \models \Pi(a_0)$ and thus there is a derivation D of $\Pi(a_0)$ in $\mathcal{A}$. Consider the ABox $\mathcal{A}_D$. By Lemma 3.2, we have the following:

1. $\mathcal{A}_D \models \Pi(a_0)$;
2. there is a homomorphism from $\mathcal{A}_D$ to $\mathcal{A}$ that is the identity on a_0;
3. $\mathcal{A}_D$ has pathwidth at most k.

We manipulate $\mathcal{A}_D$ as follows:

- restrict the degree to $|\mathcal{T}|$ by taking a subset according to Lemma 2.2;

- remove all assertions that involve an individual a that is not reachable from a_0 in $G_{\mathcal{A}}$ by a directed path.

We use $\mathcal{B}$ to denote the resulting ABox. It can be verified that Conditions 1 to 3 still hold when $\mathcal{A}_D$ is replaced with $\mathcal{B}$. In particular, this is true for Condition 1 since $\mathcal{A}_D \models \Pi(a_0)$ iff $\mathcal{A}_D \models Q_k(a_0)$ iff $\mathcal{B} \models Q_k(a_0)$ iff $\mathcal{B} \models \Pi(a_0)$. The second equivalence is easy to establish by showing how a model witnessing $\mathcal{B} \not\models Q_k(a_0)$ can be transformed into a model that witnesses $\mathcal{A}_D \not\models Q_k(a_0)$.

Choose a homomorphism h from $\mathcal{B}$ to $\mathcal{A}$ that is the identity on a_0. Then h must be surjective since otherwise, the restriction $\mathcal{A}^-$ of $\mathcal{A}$ to the individuals in the range of h would satisfy $\mathcal{A}^-, \mathcal{T} \models A_0(a_0)$, contradicting the minimality of $\mathcal{A}$. Let $a_1, \dots, a_m$ be the leaves of $\mathcal{A}$, $m \geq (n-2k-3)^{2k+3}$. For each a_i, choose a b_i with $h(b_i) = a_i$. Clearly, all individuals in $b_1, \dots, b_m$ must be distinct.

By construction, $\mathcal{B}$ is connected. Since there is a homomorphism from $\mathcal{B}$ to $\mathcal{A}$, $\mathcal{B}$ must be a DAG (directed acyclic graph). We proceed to exhaustively remove assertions from $\mathcal{B}$ as follows: whenever $r(c_1, c), r(c_2, c) \in \mathcal{B}$ with $c_1 \neq c_2$, then choose and remove one of these two assertions. Using the fact that every individual in $\mathcal{B}$ is reachable from a_0, it can be proved by induction on the number of edge removals that the obtained ABoxes

(i) remain connected and

(ii) contain the same individuals as $\mathcal{B}$, that is, edge removal never results in the removal of an individual.

Point (i) and the fact that we start from a DAG-shaped ABox means that the ABox $\mathcal{B}_t$ ultimately obtained by this manipulation is tree-shaped. By construction of $\mathcal{B}_t$, h is still a homomorphism from $\mathcal{B}_t$ to $\mathcal{A}$, $\mathcal{B}_t$ has pathwidth at most k, and the individuals $b_1, \dots, b_m$ are leaves in $\mathcal{B}_t$ (and thus $\mathcal{B}_t$ has at least $(n-2k-3)^{2k+3}$ leaves). From the former, it follows that the depth of $\mathcal{B}_t$ is at most $n(i+1)$.

Assume that $\mathcal{B}_t$ does not contain the full binary tree of depth $2k+3$ as a minor. Then by Lemma 3.26, the number of leaves of $\mathcal{B}_t$ is at most

$$(2k+3)(|\mathcal{T}|-1)^{2k+2}(n(i+1))^{2k+2},$$

which is a polynomial in n of degree $2k+2$. So if we choose n such that

$$(n-2k-3)^{2k+3} > (2k+3)(|\mathcal{T}|-1)^{2k+2}(n(i+1))^{2k+2}$$

in the beginning, this leads to a contradiction. Hence, $\mathcal{B}_t$ must contain as a minor the full binary tree of depth at least $2k+3$. But it is well-known that any such tree has pathwidth at least $k+1$, in contradiction to $\mathcal{B}_t$ having pathwidth at most k. □

We are now ready to establish the hierarchy.

Proposition 3.29. *$Q_{8\ell+13}$ is not rewritable into a linear Datalog program of width ℓ.*

Proof. Assume to the contrary of what we have to show that $Q_{8\ell+13}$ is rewritable into a linear Datalog program Π_0 of width ℓ. Let k be the diameter of Π_0. Clearly, Π_0 is also a rewriting of $Q_{8\ell+13}$ on the class of ABoxes $\mathfrak{C}_k$. We show that Π_0 can be rewritten into a linear Datalog rewriting Π' of $Q_{8\ell+13}$ of diameter $4\ell + 5$, in contradiction to Lemma 3.28.

We carry out a sequence of three rewriting steps. Informally, in the first rewriting we normalize the shape of rule bodies, in the second one we control the number of disconnected components in the rule body (or rather its restriction to the EDB relations), and in the third one we actually bound the diameter to $4\ell + 5$.

In the first step, let Π_1 be obtained from Π_0 by replacing every rule $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ in Π_0 with the set of all rules $S(\mathbf{x}') \leftarrow q(\mathbf{y}')$ that can be obtained from the original rule by consistenly identifying variables in the rule body and head such that the restriction of $q(\mathbf{y}')$ to EDB relations (that is, concept and role names in Σ) takes the form of a forest in which every tree branches at most once. This step preserves equivalence on $\mathfrak{C}_k$ since every homomorphism from the body of a rule in Π into an ABox from $\mathfrak{C}_k$ (and also to the extension of such an ABox with IDB relations) induces a variable identification that identifies a corresponding rule produced in the rewriting.

In the next step, we rewrite Π_1 into a linear Datalog program Π_2, as follows. Let $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ be a rule in Π_1 and call a variable in $q(\mathbf{y})$ *special* if it occurs in $\mathbf{x}$ or in the IDB atom in $q(\mathbf{y})$, if existent. We obtain a new rule body $q'(\mathbf{y}')$ from $q(\mathbf{y})$ in the following way:

1. remove the IDB atom (if existent), obtaining a forest-shaped rule body;
2. remove all trees that do not contain a special variable;
3. re-add the IDB atom (if existent).

In Π_2, we replace $S(\mathbf{x}) \leftarrow q(\mathbf{y})$ with $S(\mathbf{x}) \leftarrow q'(\mathbf{y}')$.

We argue that, on the class of ABoxes $\mathfrak{C}_k$, Π_2 is equivalent to Π_1. Thus, let $\mathcal{A}$ be an ABox from $\mathfrak{C}_k$ and $a \in \mathsf{ind}(\mathcal{A})$ such that $\mathcal{A} \models \Pi_2(a)$. We have to show that $\mathcal{A} \models \Pi_1(a)$. Let $q_1(\mathbf{x}_1), \dots, q_m(\mathbf{x}_m)$ be all trees that have been removed from a rule body during the construction of Π_2. Let $\mathcal{A}_i$ be $q_i(\mathbf{x}_i)$ viewed as a Σ-ABox, $1 \leq i \leq m$. Note that each $\mathcal{A}_i$ must be in $\mathfrak{C}_k$. Let $\mathcal{B}$ be the disjoint union of the ABoxes $\mathcal{A}, \mathcal{A}_1, \dots, \mathcal{A}_m$, assuming that these ABoxes do not share any individual names, and note that $\mathcal{B}$ is in $\mathfrak{C}_k$. Since $\mathcal{A} \models \Pi_2(a)$, we must have $\mathcal{B} \models \Pi_2(a)$. By construction of $\mathcal{B}$, this clearly implies $\mathcal{B} \models \Pi_1(a)$. Consequently, $\mathcal{B} \models Q_{8\ell+13}(a)$. Since answers to OMQs from ($\mathcal{EL}$, AQ) depend only on the reachable part of ABoxes, we obtain that $\mathcal{A} \models Q_{8\ell+13}(a)$, thus $\mathcal{A} \models \Pi_1(a)$ as required.

At this point, let us sum up the most important properties of the linear Datalog program Π_2: it is a rewriting of $Q_{8\ell+13}$ on $\mathfrak{C}_k$, has width at most ℓ and diameter at most k, and

($*$) the restriction of the rule body to EDB relations is a forest that consists of at most 2ℓ trees.

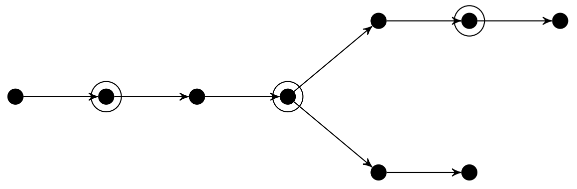

Figure 3.4: The body $q(\mathbf{y})$ of a rule from Π_2 consists of one or several such trees, where at most one variable is branching. The branching variable and special variables are circled and they divide the body into five paths $q_i(\mathbf{y}_i)$.

Note that the upper bound of 2ℓ is a consequence of the fact that, by construction of Π_2, each of the relevant trees contains at least one special variable.

We now rewrite Π_2 into a final linear Datalog program Π_3 that is equivalent to Π_2, has width at most $4\ell + 2$, and diameter at most $4\ell + 5$. Thus Π_3 is a rewriting of $Q_{8\ell+13}$ on $\mathfrak{C}_k$ of diameter $4\ell + 5$, which is a contradiction to Lemma 3.28.

It thus remains to give the construction of Π_3. Let $\rho = S(\mathbf{x}) \leftarrow q(\mathbf{y})$ be a rule in Π_2 and let $\mathbf{y}' \subseteq \mathbf{y}$ be the set of variables x that are special or a branching variable where the latter means that $q(\mathbf{y})$ contains atoms of the form $r(x, y_1)$, $s(x, y_2)$ with $y_1 \neq y_2$. Due to $(*)$, $\mathbf{y}'$ contains at most 4ℓ variables. Let $q'(\mathbf{y}')$ be the restriction of $q(\mathbf{y})$ to the variables in $\mathbf{y}'$; we can assume that each variable y from $\mathbf{y}'$ occurs in $q'(\mathbf{y}')$ since if this is not the case, we can add an atom $\top(y)$. By construction of Π_2, it can be verified that $q(\mathbf{y})$ is the union of $q'(\mathbf{y}')$ and path-shaped $q_1(\mathbf{y}_1), \dots, q_n(\mathbf{y}_n)$ such that for $1 \leq i \leq n$

- $q_i(\mathbf{y}_i)$ contains only EDB atoms,
- each $q_i(\mathbf{y}_i)$ contains at most two variables from $\mathbf{y}'$ and each such variable is an end point of the path, and
- the queries $q_1(\mathbf{y}_1), \dots, q_n(\mathbf{y}_n)$ only share variables from $\mathbf{y}'$.

The structure of $q(\mathbf{y})$ is illustrated in Figure 3.4. We thus find linear Datalog programs $\Gamma_1, \dots, \Gamma_n$ that are at most binary, of width at most two and diameter at most three such that for any Σ-ABox $\mathcal{A}$ and $\mathbf{a} \subseteq \mathsf{ind}(\mathcal{A})$, $\mathcal{A} \models \Gamma_i(\mathbf{a})$ if and only if there is a homomorphism h_i from $q_i(\mathbf{y}_i)$ to $\mathcal{A}$ such that $h_i(\mathbf{y}_i) = \mathbf{a}$. Let the goal relations of $\Gamma_1, \dots, \Gamma_n$ be $G_1, \dots, G_n$ and assume that G_i occurs in Γ_i only once, in a rule head $G_i(\mathbf{x}_i)$. We assume w.l.o.g. that the programs $\Gamma_1, \dots, \Gamma_n$ do not share variables or IDB relations, and neither do they share variables or IDB relations with Π_2. In Π_3, we replace $\rho = S(\mathbf{x}) \leftarrow q(\mathbf{y})$ with the following rules:

- for any rule $P(\mathbf{w}) \leftarrow p(\mathbf{z})$ in Γ_1 where $p(\mathbf{z})$ contains only EDB atoms, the rule $X^P_\rho(\mathbf{y}', \mathbf{w}) \leftarrow q'(\mathbf{y}') \wedge p(\mathbf{z})$;
- for any rule $P(\mathbf{w}) \leftarrow p(\mathbf{z})$ in Γ_i, $1 \leq i \leq n$, where $p(\mathbf{z})$ contains the IDB atom $R(\mathbf{u})$, the rule $X^P_\rho(\mathbf{y}', \mathbf{w}) \leftarrow X^R_\rho(\mathbf{y}', \mathbf{u}) \wedge p(\mathbf{z})$;
- for any rule $P(\mathbf{w}) \leftarrow p(\mathbf{z})$ in Γ_i, $1 < i \leq n$, where $p(\mathbf{z})$ contains only EDB atoms, the rule $X^P_\rho(\mathbf{y}', \mathbf{w}) \leftarrow X^{G_{i-1}}_\rho(\mathbf{y}', \mathbf{x}_{i-1}) \wedge p(\mathbf{z})$;
- the rule $S(\mathbf{x}) \leftarrow X^{G_n}_\rho(\mathbf{y}', \mathbf{x}_n)$,

where the goal relations of $\Gamma_1, \dots, \Gamma_n$ become standard (non-goal) IDB relations. It can be verified that Π_3 is as required. □

3.6 Decidability and Complexity

We study the meta problems that emerge from the results in the previous sections such as deciding whether a given OMQ is in NL, PTime-hard, or rewritable into linear Datalog. We show that all these problems are ExpTime-complete. Apart from applying and adapting known lower and upper bounds, the central ingredient is giving a single exponential time decision procedure for deciding whether an OMQ from $(\mathcal{EL}, \text{conCQ})$ has the ability to simulate PSA. We start with lower bounds, which hold already for $(\mathcal{EL}, \text{AQ})$.

Theorem 3.30. *Given an OMQ $Q \in (\mathcal{EL}, \text{AQ})$, the following problems are ExpTime-hard:*

(1) Is Q FO rewritable?

(2) Is Q rewritable into linear Datalog?

(3) Is $\text{EVAL}(Q) \in AC^0$?

(4) Is $\text{EVAL}(Q) \in NL$? (unless NL = PTime)

(5) Is $\text{EVAL}(Q)$ NL-hard?

(6) Is $\text{EVAL}(Q)$ PTime-hard?

Proof. ExpTime-hardness of (1) is proved in (the appendix of) [BLW13]. By our Theorem 3.20, (1) and (3) are equivalent, so (3) is also ExpTime-hard.

For (2), (4), (5) and (6), we analyse the mentioned hardness proof from [BLW13] a little closer. The proof is by a reduction from the word problem of a polynomially space bounded alternating Turing machine (ATM) M that solves an ExpTime-complete problem. The reduction exhibits a polynomial time algorithm that constructs, given an input w to M, an OMQ $Q = (\mathcal{T}, \Sigma, B(x)) \in (\mathcal{EL}, \text{AQ})$ such that Q is not FO rewritable if and only if M accepts w. A careful inspection of the construction of Q and of the "$\Leftarrow$" part of the proof reveals that

($*$) if M accepts w, then Q is unboundedly branching, thus (by Lemma 3.11 and Theorem 3.22) not linear Datalog rewritable, PTime-hard, and not in NL (unless NL = PTime) and (by Theorem 3.20 and since no PTime-hard problem can be in AC^0) NL-hard.

If M does not accept w, then FO rewritability of Q implies that Q is

- in AC^0 and thus in NL and neither NL-hard nor PTime-hard;
- linear Datalog rewritable (because every FO rewritable OMQ from $(\mathcal{EL}, \text{AQ})$ is rewritable into a UCQ [BLW13]).

The stated hardness results for (2), (4), (5) and (6) follow. □

The following theorem summarizes the corresponding upper bounds.

Theorem 3.31. *Given $Q \in (\mathcal{EL}, \text{CQ})$, the following properties can be decided in ExpTime:*

(1) Is Q FO rewritable?

(2) Is Q rewritable into linear Datalog?

(3) Is $\textsc{eval}(Q) \in AC^0$*?*

(4) Is $\textsc{eval}(Q) \in$ *NL? (unless NL = PTime)*

(5) Is $\textsc{eval}(Q)$ *NL-hard?*

(6) Is $\textsc{eval}(Q)$ *PTime-hard? (unless NL = PTime)*

In [Bie+16], it was shown that (1) is in ExpTime. By Theorem 3.20, the same algorithm decides (3) and (5). By Theorem 3.22, the remaining (2), (4) and (6) come down to a single decision problem. We concentrate on deciding (6). We first argue that it suffices to decide (6) for OMQs from $(\mathcal{EL}, \text{conCQ})$, that is, to restrict our attention to connected CQs.

Let $Q = (\mathcal{T}, \Sigma, q) \in (\mathcal{EL}, \text{CQ})$. To decide whether $\textsc{eval}(Q)$ is PTime-hard (unless NL = PTime), we can first check whether Q is empty. This can be done in ExpTime [Bie+16]) and an empty OMQ is clearly not PTime-hard. Otherwise, we make Q non-redundant (see Section 3.4) by exhaustively removing Boolean MCCs that cause non-redundancy. This can also be done in exponential time since containment of OMQs from $(\mathcal{EL}, \text{CQ})$ is in ExpTime [Bie+16]. The resulting OMQ $Q' = (\mathcal{T}, \Sigma, q')$ is equivalent to Q and as seen in the proof of Theorem 3.22, $\textsc{eval}(Q')$ is PTime-hard if and only if there is an MCC q'_i of q' such that $(\mathcal{T}, \Sigma, q'_i) \in (\mathcal{EL}, \text{conCQ})$ is PTime-hard.

Therefore, it remains to show how (6) can be decided in ExpTime for OMQs Q from $(\mathcal{EL}, \text{conCQ})$. For such Q, it follows from Lemmas 3.11, 3.15, 3.16, and 3.17 and Theorem 3.9 that (6) is equivalent to deciding whether Q has the ability to simulate $\textsc{psa}$. In the remainder of this section, we reduce the question whether a given OMQ $Q \in (\mathcal{EL}, \text{conCQ})$ has the ability to simulate $\textsc{psa}$ to the (non-)emptiness problem of TWAPA (introduced in Section 2.9), which is in ExpTime. In fact, we construct a TWAPA that accepts precisely those (encodings of) pseudo tree-shaped ABoxes that witness the ability to simulate $\textsc{psa}$ and then check non-emptiness.

Encoding pseudo tree-shaped ABoxes. To check the ability to simulate $\textsc{psa}$ using TWAPAs, we build one TWAPA $\mathfrak{A}_{t_0,t_1}$ for every pair (t_0, t_1) of $\mathcal{T}$-types. An input tree for the TWAPA encodes a tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ of a pseudo tree-shaped ABox $\mathcal{A}$ of core size at most $|q|$, a tuple $\mathbf{a}$ from the core and three distinguished individuals b, c and d. The TWAPA $\mathfrak{A}_{t_0,t_1}$ should accept a tree that encodes $(\mathcal{A}, \mathbf{a}, b, c, d)$ if and only if $t_0, t_1, \mathcal{A}, \mathbf{a}, b, c$ and d witness the ability to simulate $\textsc{psa}$ according to Definition 3.13. The ExpTime decision procedure is obtained by checking whether at least one of the (exponentially many) $\mathfrak{A}_{t_0,t_1}$ accepts a non-empty language.

We encode tuples $(\mathcal{A}, \mathbf{a}, b, c, d)$ as finite $(|\mathcal{T}| \cdot |q|)$-ary $\Gamma_\varepsilon \cup \Gamma_N$-labeled trees, where Γ_ε is the alphabet used for labeling the root node and Γ_N is for non-root nodes. These alphabets are different because the root of a tree encodes the entire core of a pseudo tree-shaped ABox whereas each non-root node represents a single non-core individual.

Let $\mathsf{C}_\mathsf{core} \subseteq \mathsf{N_I}$ be a fixed set of size $|q|$. Define Γ_ε to be the set of all tuples $(\mathcal{B}, \mathbf{a})$, where $\mathcal{B}$ is a Σ-ABox over C_core and $\mathbf{a}$ a tuple of length $\mathsf{ar}(q)$ from $\mathsf{ind}(\mathcal{B})$. Let ROL be the set of roles that appear in $\mathcal{T}$ or Σ and let CN by the set of all concept names that appear in $\mathcal{T}$ or Σ. Let $S = \mathsf{ROL} \cup \mathsf{CN} \cup \mathsf{C}_{\mathsf{core}} \cup \{b, c, d\}$. The alphabet Γ_N is defined to be the set of all subsets of S that contain exactly one element from ROL, at most one element from $\mathsf{C}_{\mathsf{core}}$ and at most one element of $\{b, c, d\}$. We call a $(\Gamma_\varepsilon \cup \Gamma_N)$-labeled tree (T, L) *proper* if

- $L(\varepsilon) \in \Gamma_\varepsilon$ and $L(x) \in \Gamma_N$ for all $x \neq \varepsilon$,
- $L(x)$ contains an element of $\mathsf{C}_{\mathsf{core}}$ if and only if x is a child of ε,
- there is exactly one node $x_b \in T$ with $b \in L(x_b)$, exactly one node $x_c \in T$ with $c \in L(x_c)$ and exactly one node $x_d \in T$ with $d \in L(x_d)$,
- the nodes x_c and x_d are incomparable descendants of x_b,
- the nodes ε, x_b, x_c and x_d have pairwise distance more than $|q|$ from each other.

A proper tree (T, L) encodes a tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ in the following way. If $L(\varepsilon) = (\mathcal{B}, \mathbf{a})$, then

$$\begin{array}{rcl}\mathcal{A} & = & \mathcal{B} \cup \{A(x) \mid A \in L(x), x \neq \varepsilon\} \\ & & \cup\ \{r(a, x) \mid \{a, r\} \subseteq L(x) \text{ with } a \in \mathsf{C}_{\mathsf{core}}\} \\ & & \cup\ \{r(x, y) \mid r \in L(y), y \text{ is a child of } x, x \neq \varepsilon\}\end{array}$$

with x_b replaced with b, x_c with c, and x_d with d. It is easy to see that there is a TWAPA $\mathfrak{A}_{\mathsf{proper}}$ that accepts a $(\Gamma_\varepsilon \cup \Gamma_N)$-labeled tree if and only if it is proper.

From now on, let t_0 and t_1 be fixed. We construct the TWAPA $\mathfrak{A}_{t_0,t_1}$ as the intersection of $\mathfrak{A}_{\mathsf{proper}}$ and TWAPAs $\mathfrak{A}_1, \ldots, \mathfrak{A}_6$ where each $\mathcal{A}_i$ accepts a proper input tree (T, L) if and only if the tuple $(\mathcal{A}, \mathbf{a}, b, c, d)$ encoded by (T, L) satisfies Condition (i) from Definition 3.13. We make sure that all $\mathfrak{A}_i$ can be constructed in exponential time and have only polynomially many states in the size of Q.

Derivation of concept names. Before describing any of the $\mathfrak{A}_i$ in detail, we describe one capability of TWAPAs that most of the $\mathfrak{A}_i$ will make use of, namely to check whether a certain concept name is derived at a certain individual. We thus construct a TWAPA $\mathfrak{A}_{\mathsf{derive}}$ with states $S_{\mathsf{derive}} =$

$$\{d_A \mid A \in \mathsf{CN}\} \cup \{d_A^a \mid A \in \mathsf{CN} \wedge a \in \mathsf{C}_{\mathsf{core}}\} \cup \{d_r \mid r \in \mathsf{ROL}\} \cup \{d_a \mid a \in \mathsf{C}_{\mathsf{core}}\}$$

such that

- if $\mathfrak{A}_{\mathsf{derive}}$ is started on a proper input tree encoding $(\mathcal{A}, \mathbf{a}, b, c, d)$ from a configuration (a, d_A), then it accepts if and only if $\mathcal{T}, \mathcal{A} \models A(a)$;
- if $\mathfrak{A}_{\mathsf{derive}}$ is started on a proper input tree encoding $(\mathcal{A}, \mathbf{a}, b, c, d)$ from a configuration (d_A^a, ε), then it accepts if and only if $\mathcal{T}, \mathcal{A} \models A(a)$.

By Lemma 2.4, $\mathcal{T}, \mathcal{A} \models A(a)$ if and only if there is a derivation tree for $A(a)$. We give the straightforward construction of $\mathfrak{A}_{\mathsf{derive}}$, that checks for the existence of a derivation tree of $A(a)$. For brevity, let $\ell = |\mathcal{T}| + |q|$. Let $\sigma \in \Gamma_N$ not contain an element of $\mathsf{C}_{\mathsf{core}}$

and let r be the unique role name in σ. If $A \in \sigma$ or $\top \sqsubseteq A \in \mathcal{T}$, we set $\delta(d_A, \sigma) = \text{true}$. Otherwise, set

$$\delta(d_A, \sigma) = \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i} \Big) \vee \Big(\bigvee_{\exists s.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_s) \Big) \vee \Big(\bigvee_{\exists r^-.B \sqsubseteq A} \langle -1 \rangle d_B \Big)$$

Now let $\sigma \in \Gamma_N$ contain $a \in \mathsf{C}_{\text{core}}$ and let again r be the unique role name in σ. If $A \in \sigma$ or $\top \sqsubseteq A \in \mathcal{T}$, we set $\delta(d_A, \sigma) = \text{true}$. Otherwise, set

$$\delta(d_A, \sigma) = \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i} \Big) \vee \Big(\bigvee_{\exists s.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_s) \Big) \vee \Big(\bigvee_{\exists r^-.B \sqsubseteq A} \langle -1 \rangle d_B^a \Big)$$

Next, let $\sigma = (\mathcal{B}, \mathbf{a}) \in \Gamma_\varepsilon$. If $A(a) \in \mathcal{B}$ or $\top \sqsubseteq A \in \mathcal{T}$, we set $\delta(d_A^a, \sigma) = \text{true}$. Otherwise, set

$$\delta(d_A^a, \sigma) = \Big(\bigvee_{\mathcal{T} \models A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A} \bigwedge_{i=1}^{n} \langle 0 \rangle d_{A_i}^a \Big) \vee \Big(\bigvee_{\exists s.B \sqsubseteq A \in \mathcal{T}} \bigvee_{i=1}^{\ell} \langle i \rangle (d_B \wedge d_s \wedge d_a) \Big) \vee \Big(\bigvee_{\exists s.B \sqsubseteq A, s(a,b) \in \mathcal{B}} \langle 0 \rangle d_B^b \Big) \vee \Big(\bigvee_{\exists s^-.B \sqsubseteq A, s(b,a) \in \mathcal{B}} \langle 0 \rangle d_B^b \Big)$$

Finally, let $\sigma \in \Gamma_N$ and $a \in \mathsf{C}_{\text{core}}$. Set $\delta(d_a, \sigma) = \text{true}$ if $a \in \sigma$ and $\delta(d_a, \sigma) = \text{false}$ if $a \notin \sigma$.

Construction of $\mathfrak{A}_1$. This TWAPA checks whether $\mathcal{A} \models Q(\mathbf{a})$. Since by Condition 5 of the ability to simulate PSA, every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close, we only need to check whether $\mathcal{A} \models Q(\mathbf{a})$ via a core close homomorphism. The existence of such a homomorphism, in turn, depends only on the core of $\mathcal{A}$ and on which concept names are derived at core individuals. If, in fact, h is a core close homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, then h hits at least one core individual or an element in an anonymous subtree below a core individual. Of course, h might also hit individuals and anonymous elements outside the core. However, outside the core $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is tree-shaped. Take any $z \in \mathsf{var}(q)$ such that $h(z)$ lies outside of the core. Then there is a unique $a \in \mathsf{C}_{\text{core}}$ such that $h(z)$ lies in a tree below a. Since q is connected, there is an atom $r(x, y) \in q$ such that $h(x) = a$, $h(y)$ lies in the tree below a, and $z \in \mathsf{reach}(x, y)$. But then C_q contains $q|_{\mathsf{reach}(x,y)}$ viewed as an $\mathcal{EL}$-concept C, $\mathcal{T} \models C \sqsubseteq A_C$, and $A_C(a) \in \mathcal{U}_{\mathcal{A},\mathcal{T}}$. Thus, the concept names A_C derived at core individuals completely represent the restriction of h to the variables in q that are mapped to outside the core.

Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox with core $\mathcal{B}$, $\mathsf{ind}(\mathcal{B}) \subseteq \mathsf{C}_{\text{core}}$, and $\mathbf{a}$ a tuple from C_{core}. If $\mathcal{A} \models Q(\mathbf{a})$, then the set $\{A(a) \mid a \in \mathsf{C}_{\text{core}} \text{ and } a \in A^{\mathcal{U}_{\mathcal{A},\mathcal{T}}}\}$ is a *completion for* $(\mathcal{B}, \mathbf{a})$. Let $\mathsf{Comp}(\mathcal{B}, \mathbf{a})$ be the set of all completions for $(\mathcal{B}, \mathbf{a})$. Using the arguments above, one can show the following.

Lemma 3.32. *Let $\mathcal{A}$ be a pseudo tree-shaped Σ-ABox with core $\mathcal{B}$, $\mathsf{ind}(\mathcal{B}) \subseteq \mathsf{C}_{\mathsf{core}}$, and* $\mathbf{a}$ *a tuple from $\mathsf{C}_{\mathsf{core}}$. Then, $\mathcal{A} \models Q(\mathbf{a})$ if and only if there is an $M \in \mathsf{Comp}(\mathcal{B}, \mathbf{a})$ such that $\mathcal{A} \models A(a)$ for all $A(a) \in M$.*

Each set $\mathsf{Comp}(\mathcal{B}, \mathbf{a})$ has at most exponentially many completions and can be computed in exponential time. Moreover, there are only exponentially many choices for $(\mathcal{B}, \mathbf{a})$. The strategy of $\mathfrak{A}_1 = (S_{\mathsf{derive}} \cup \{s_0\}, \Gamma_\varepsilon \cup \Gamma_N, \delta, s_0, c)$ is as follows: Guess a match completion set M for Q regarding $(\mathcal{B}, \mathbf{a})$, where $(\mathcal{B}, \mathbf{a}) \in \Gamma_\varepsilon$ is the label of the root of the input tree and then check whether all $A(a) \in M$ can be derived in $\mathcal{A}$. The parity condition c assigns 1 to every state, so precisely the finite runs are accepting. The transition function δ for states in S_{derive} is defined as before, and additionally we set

$$\delta(s_0, (\mathcal{B}, \mathbf{a})) = \bigvee_{M \in \mathsf{Comp}_Q^{(\mathcal{B}, \mathbf{a})}} \bigwedge_{A(a) \in M} d_A^a.$$

Construction of $\mathfrak{A}_2$. This TWAPA checks that $t_1 = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(b) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(c) = \mathsf{tp}_{\mathcal{A},\mathcal{T}}(d)$. Using $\mathfrak{A}_{\mathsf{derive}}$ and its dualization, this is straightforward: send a copy to the nodes in the input tree that are marked with b, c, and d, and then use $\mathfrak{A}_{\mathsf{derive}}$ to make sure that all $A \in t_1$ are derived there and the dual of $\mathfrak{A}_{\mathsf{derive}}$ to make sure that no $A \notin t_1$ is derived there.

Construction of $\mathfrak{A}_3$. This TWAPA checks that $\mathcal{A}_b \cup t_0(b), \mathcal{T} \not\models q(\mathbf{a})$. It is constructed in the same way as $\mathfrak{A}_1$, but using $\mathcal{A}_b \cup t_0(b)$ instead of $\mathcal{A}$ for defining completions and modifying $\mathfrak{A}_{\mathsf{derive}}$ to that it assumes all concept names from t_0 to be true at the node of the input tree marked with b and disregards the subtree below. Also, we complement the constructed TWAPA at the end.

Construction of $\mathfrak{A}_4$. Similar to $\mathcal{A}_3$.

Construction of $\mathfrak{A}_5$. This TWAPA checks that every homomorphism from q to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is core close and this condition is only required if q is Boolean. The condition is always true when q is not treeifiable, since every homomorphism from a query that is not treeifiable into $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ hits the core. Thus, if Q is not Boolean or not treeifiable, we define $\mathfrak{A}_5$ to be the TWAPA that accepts every input. If Q is Boolean and treeifiable, we define a TWAPA $\mathfrak{A}_5'$ that checks the negation of Condition 5 and then define $\mathfrak{A}_5$ to be the complement of $\mathfrak{A}_5'$. Let C_q be the $\mathcal{EL}$-concept that corresponds to q^{tree}. The TWAPA $\mathfrak{A}_5'$ has to check whether C_q is derived at any non-core individual a or at an anonymous individual below a non-core individual. To check whether C_q gets derived at an anonymous individual, define M_Q be the set of all $\mathcal{T}$-types t with $t(a), \mathcal{T} \models \exists x\, C_q(x)$. The set M_Q can be computed in exponential time. Now $\mathfrak{A}_5'$ guesses a non-core individual a and $t \in M_Q$ and checks that $\mathsf{tp}_{\mathcal{A},\mathcal{T}}(a) = t$ using $\mathfrak{A}_{\mathsf{derive}}$ and its dualization.

Construction of $\mathfrak{A}_6$. Condition 6 is only required when q is Boolean. If q is Boolean, this TWAPA checks that b, c and d all have the same ancestor path up to length $|q|$. The idea is to guess an ancestor path $r_1 r_2 \ldots r_{|q|}$ up front and then verify that b, c and d all have this ancestor path. To achieve this using only polynomially many states, the guessed path is not stored in a single state. Instead, we use $|q|$ copies of the automaton, the i-th copy guessing states of the form $s_{i,r}$ which stands for $r_i = r$. This copy then further spawns into three copies that visit the nodes labeled b, c, and d, travels upwards from there $n - i$ steps, and checks that the node label there contains r.

3.7 Conclusion

We have established a complexity trichotomy between AC^0, NL, and PTime for OMQs from $(\mathcal{EL}, \text{CQ})$. We have also proved that linear Datalog rewritability coincides with OMQ evaluation in NL and that deciding all these (and related) properties is ExpTime complete with the lower bounds applying already to $(\mathcal{EL}, \text{AQ})$.

There are several natural directions in which our results can be generalized. One is to transitions from CQs to unions of CQs (UCQs), that is, to consider the OMQ language $(\mathcal{EL}, \text{UCQ})$. We conjecture that this generalization is not difficult and can be achieved by replacing CQs with UCQs in all of our proofs; where we work with connected CQs, one would then work with UCQs in which every CQ is connected. In fact, we only refrained from doing so since it makes all proofs more technical and distracts from the main ideas.

An important direction for future work is to extend our analysis to more expressive ontology languages. A natural choice would be $\mathcal{ELI}$, that is, to add inverse roles. Even the case of $(\mathcal{ELI}, \text{AQ})$ appears to be challenging. In the final section of this chapter, we elaborate on this extension.

3.8 Towards a Classification for $(\mathcal{ELI}, \text{AQ})$

When trying to generalize the results from $(\mathcal{EL}, \text{AQ})$ to $(\mathcal{ELI}, \text{AQ})$, one notices several difficulties and characterizations from the $\mathcal{EL}$ case fail to hold in the $\mathcal{ELI}$ case. In fact, it can be seen that a complexity classification of $(\mathcal{ELI}, \text{AQ})$ is equivalent to a complexity classification of all CSPs that have tree obstructions. In this section we point out these difficulties, give counter examples to theorems from this chapter when $\mathcal{EL}$ is replaced with $\mathcal{ELI}$, and formulate a conjecture on the complexity classification in $(\mathcal{ELI}, \text{AQ})$.

LogSpace and Symmetric Datalog

First, one notices that $(\mathcal{ELI}, \text{AQ})$ contains OMQs that are complete for LogSpace. The following OMQ is easily seen to be equivalent (under FO reductions) to the reachability problem in undirected graphs (ureach), which is known to be LogSpace complete [Rei04].

Example 3.33. *Let $Q = (\mathcal{T}, \Sigma, A(x))$ with $\mathcal{T} = \{\exists r.A \sqsubseteq A,\ \exists r^-.A \sqsubseteq A\}$ and $\Sigma = \{r, A\}$. Intuitively, this OMQ asks for an undirected r-path to an individual labeled with A. The reductions between* eval(Q) *and* ureach *are straightforward: An undirected graph $G = (V, E)$ with start node s and target node t is encoded as the ABox $\{r(a, b) \mid \{a, b\} \in E\} \cup A(s)$ and we ask whether $\mathcal{A} \models Q(t)$. Conversely, a pair $(\mathcal{A}, t)$ of a Σ-ABox $\mathcal{A}$ and an individual*

$t \in \mathsf{ind}(\mathcal{A})$ is translated into the instance $G = (V, E, s, t)$ with $V = \mathsf{ind}(\mathcal{A}) \cup \{s\}$ and $E = \{\{a, b\} \mid r(a, b) \in \mathcal{A}\} \cup \{\{s, a\} \mid A(a) \in \mathcal{A}\}$.

Using a variation of the techniques from Section 3.2 and the technique of *transfer sequences* from [Bie+16], it should not be too hard to establish a dichotomy between AC^0 and LogSpace in $(\mathcal{ELI}, \mathrm{AQ})$. This dichotomy is not surprising, since it also holds for CSPs, as established in [LT09].

Furthermore, there is another natural fragment of Datalog, even of linear Datalog, called *symmetric Datalog*, introduced in [ELT07], that we conjecture to coincide with LogSpace data complexity in the same way that linear Datalog corresponds to NL data complexity (which we conjecture to hold still for $(\mathcal{ELI}, \mathrm{AQ})$). A linear Datalog program is called *symmetric* if for every rule

$$P_1(\mathbf{x}_1) \leftarrow P_2(\mathbf{x}_2) \land R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)$$

with IDBs P_1 and P_2, the program also contains the rule

$$P_2(\mathbf{x}_2) \leftarrow P_1(\mathbf{x}_1) \land R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)\,.$$

It is proved in [ELT07] and easy to see that evaluating a symmetric Datalog program is in LogSpace regarding data complexity, by reducing the problem to ureach. There are indications that for CSPs, the opposite might hold, that is, that all CSPs in LogSpace can be expressed in symmetric Datalog [ELT08].

The intuition says that OMQs asking for undirected reachability, like the OMQ in Example 3.33, are LogSpace-complete, while OMQs that ask for directed reachability are NL-complete. But what if we try to construct OMQs that have features of both directed and undirected reachability? Can we construct an OMQ with data complexity strictly inbetween LogSpace and NL? Consider the following example, which will be the running example for the rest of this section.

Example 3.34. *Let $Q = (\mathcal{T}, \Sigma, B(x))$ with $\Sigma = \{r, s, t, A\}$ and*

$$\begin{aligned} \mathcal{T} = \{&\exists t.A \sqsubseteq A, \exists t^-.A \sqsubseteq A, \\ &\exists s.A \sqsubseteq B, \\ &\exists r.B \sqsubseteq B, \exists r^-.B \sqsubseteq B\}\,. \end{aligned}$$

Intuitively, this OMQ asks for an undirected r-path to an individual which has an outgoing s-edge to an individual that has an undirected t-path to an individual labeled A. While the t-path and the r-path are undirected, the s-edge has to be directed. Even though mixing features of directed and undirected reachability, this OMQ is both in LogSpace and also rewritable into symmetric Datalog.

It is relatively easy to see that eval(Q) is in LogSpace. To check whether $\mathcal{A} \models Q(a)$, a LogSpace algorithm can iterate over all s-edges and then check for an undirected r-path from a to the start of the s-edge and then check for an undirected t-path from the end of the s-edge to an individual labeled A. However, it is not so easy to see how Q can

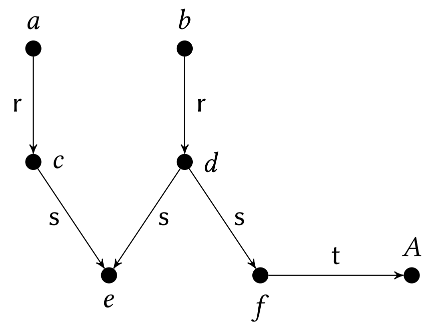

Figure 3.5: The symmetric Datalog program Π derives not only goal(b), but also goal(a), whereas $\mathcal{A} \not\models Q(a)$. The reason is that after deriving $P_B(d)$, the program does not 'remember' that the IDB fact used to derive $P_B(d)$ was $P_A(f)$ and not $P_A(e)$. Thus, the rule (∗) can be used to derive $P_A(e)$, and then we obtain $P_B(c)$, $P_B(a)$ and goal(a).

be rewritten into a symmetric Datalog program. The naive way of translating $\mathcal{T}$ into a linear Datalog program and then closing the rules under symmetry does not work. This would yield the following program Π:

$$\begin{array}{rcll}
P_A(x) & \leftarrow & A(x) & \\
P_A(x) & \leftarrow & P_A(y) \land t(x,y) & \\
P_A(y) & \leftarrow & P_A(x) \land t(x,y) & \\
P_B(x) & \leftarrow & P_A(y) \land s(x,y) & \\
P_A(y) & \leftarrow & P_B(x) \land s(x,y) & (*) \\
P_B(x) & \leftarrow & P_B(y) \land r(x,y) & \\
P_B(y) & \leftarrow & P_B(x) \land r(x,y) & \\
\mathrm{goal}(x) & \leftarrow & P_B(x) &
\end{array}$$

The rule labeled with (∗) is the rule added to close the program under symmetry. But precisely this rule causes problems, as seen in Figure 3.5.

The question remains whether Q is rewritable into symmetric Datalog. In fact, we can find a simple *stratified symmetric Datalog* program. A linear Datalog program Π is called *stratified symmetric*, if there is a function d that assigns a natural number, called the *degree*, to every IDB predicate such that for every rule $P_1(\mathbf{x}_1) \leftarrow P_2(\mathbf{x}_2) \land R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)$ from Π with IDB predicates P_1 and P_2, either

- $d(P_1) = d(P_2)$ and the symmetric rule $P_2(\mathbf{x}_2) \leftarrow P_1(\mathbf{x}_1) \land R_1(\mathbf{y}_1) \land \ldots \land R_n(\mathbf{y}_n)$ is also in Π or
- $d(P_1) > d(P_2)$.

One can then prove the following:

Lemma 3.35. *For every stratified symmetric Datalog program, there exists an equivalent symmetric Datalog program.*

In fact, the rule ($*$) can be removed from the program Π above and defining the degrees $d(P_A) = 1$, $d(P_B) = 2$ and $d(\text{goal}) = 3$. Using Lemma 3.35, one can settle the status of our example OMQ Q to be rewritable into symmetric Datalog.

We do not give a full proof of Lemma 3.35 here, but just an idea: Let Π be a stratified symmetric Datalog program. By some simple syntactic manipulations, we can transform Π into a program of the following form: The degrees of IDBs in Π are $\{1, \dots, n\}$ for some $n \in \mathbb{N}$. The goal predicate is the only IDB of degree n. There are no rules with the goal predicate in the body. All non-recursive rules have an IDB of degree 1 in the head. If $P_1(\mathbf{x}) \leftarrow P_2(\mathbf{y}) \wedge \dots$ is a recursive rule and $d(P_1) > d(P_2)$, then $d(P_1) = d(P_2) + 1$. Let T_j be the set of all IDBs from Π with degree j. The set of IDBs of the symmetric Datalog program Π' that should be equivalent to Π is $(\prod_{j=1}^{n} T_j) \times \{1, \dots, n\} \cup \{\text{goal}\}$. The arity of an IDB $(P_1, P_2, \dots, P_n, j)$ is the sum of the arities of $P_1, P_2, \dots, P_n$. Intuitively, such an IDB remembers not just one IDB fact from Π, but also a part of the history how this IDB fact was derived in Π: Whenever we go from an IDB fact of degree i to an IDB fact of degree $i + 1$, we remember the fact of degree i and the integer in the last component remembers which IDB is currently active. Using these IDBs, it is possible to construct a program that deals with the problem shown in Figure 3.5.

Besides the OMQ from Example 3.34, we also considered more complicated examples. The analysis of these examples showed that every of these OMQs was either rewritable into (stratified) symmetric Datalog or NL-hard. We thus conjecture that ($\mathcal{ELI}$, AQ) has a dichotomy between LogSpace and NL and that symmetric Datalog coincides with LogSpace complexity. The same is actually conjectured for CSPs, but it seems difficult to approach the dichotomy between LogSpace and NL without first solving the NL versus PTime case. In this context, it is interesting to point out that for CSPs, the following conditional result is known [Kaz15]: if rewritability into linear Datalog coincides with NL, then rewritability into symmetric Datalog coincides with LogSpace.

NL versus PTime

Lifting our dichotomy between NL and PTime to ($\mathcal{ELI}$, AQ) is also non-trivial. In fact, we give below an example which shows that unbounded branching no longer coincides with bounded pathwidth and thus our proof strategy, which uses unbounded branching in central places, has to be revised.

Example 3.36. *Consider the OMQ $Q = (\mathcal{T}, \Sigma, A(x)) \in (\mathcal{ELI}, \text{AQ})$ with $\Sigma = \{r, s, B, E, L\}$ and*

$$\begin{aligned}\mathcal{T} = \{&B \sqsubseteq M_1, \exists s.M_1 \sqsubseteq M_1, \exists r M_1 \sqsubseteq M_1', \exists s^- M_1' \sqsubseteq M_2,\\ &\exists r^- M_2 \sqsubseteq M_2, M_2 \sqcap L \sqsubseteq M_1, M_2 \sqcap E \sqsubseteq A, \exists s.A \sqsubseteq A\}\,.\end{aligned}$$

Q is unboundedly branching, as witnessed by the ABox in Figure 3.8 and generalizations thereof to arbitrary depth. A derivation of the query starts at B, the beginning marker, then it uses markers M_1 and M_2 to visit all the leafs from left to right in sequence, until it reaches E, the end marker, to derive the queried concept name A.

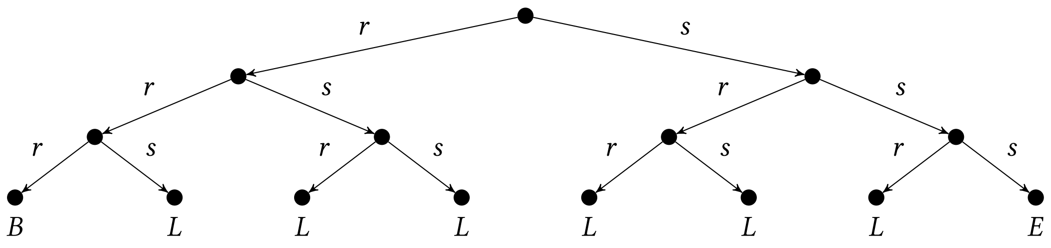

Figure 3.6: An ABox $\mathcal{A}$ with $\mathcal{A} \models Q(a)$, where a is the root of $\mathcal{A}$ and Q is the OMQ from Example 3.36.

At the same time, Q is rewritable into linear Datalog and thus $\textsc{eval}(Q) \in$ *NL, showing that unbounded branching and* PTime*-hardness no longer coincide.*

This example shows that unbounded branching alone is not sufficient for PTime-hardness and one has to find a different approach to go from unbounded pathwidth to PTime-hardness. Another task is to find a suitable $\mathcal{ELI}$ version of the ability to simulate PSA, because it is essential in the proof of Lemma 3.16 that we deal with the directed derivation of $\mathcal{EL}$.

Conjecture

Based on our observations, we conjecture that every OMQ from $(\mathcal{ELI}, \text{AQ})$ is complete for either AC^0, LogSpace, NL, or PTime. We also believe that LogSpace-completeness coincides with rewritability into symmetric Datalog, and that NL-completeness still coincides with rewritability into linear Datalog.

4 Query-by-Example for Expressive Horn Description Logic Ontologies

In this chapter, we introduce the query-by-example (QBE) paradigm for query answering in the presence of ontologies. Intuitively, QBE permits non-expert users to explore and understand knowledge bases by providing examples of data from the ABox they want and examples they do not want, which the system then generalizes into a query that, evaluated as an OMQ, returns all positive but none of the negative examples. We focus on knowledge bases with ontologies formulated in $\mathcal{ELI}$ and Horn-$\mathcal{ALC}$ and (unions of) conjunctive queries.

In recent times, the success of OBDA has led not only to the development of a vast amount of foundational results, but also of optimized systems, so-called *ontology-enriched systems (OES)* which are used in real-life scenarios, see e.g. [RKZ13; Kha+15; Cal+16; Hov+17] and references therein. For instance, the OES Ontop is currently being used to access exploration data generated by the petroleum company Equinor ASA (formerly Statoil) [Kha+15]. In these OES, users access the data through queries usually formulated in powerful query languages such as conjunctive or path queries. Unfortunately, in real life, casual non-expert users are often not able to specify queries using these formalisms (e.g., Statoil geologists [Hov+17]), clearly hampering the usability of OES.

The same problem even occurs in the context of traditional databases (without ontology), when a non-expert user tries to become familiar with a big database using a complicated schema. In this context, an alternative approach for querying was proposed to alleviate this problem: *query-by-example (QBE)*, where users give positive and negative examples from the database which the system should reverse-engineer into a query conforming with the examples [Zlo75]. When working with big data, this problem becomes even more relevant, since working with a number of different data sources using different schemas makes it more difficult to formulate the correct query [Bon+14; Mot+17]. Thus, the QBE approach has lately gained a lot of attention; indeed, even expert users might find it useful to explore the data using this paradigm as follows: They provide positive and negative examples of data, the system generalizes these into a query or reports that there is no such query. If there such a query exists, it might be the case that it is still not equivalent to the intended query. The user then adds more positive or negative examples, until the intended query is obtained. As a result, QBE has been investigated for different query languages and data representations, e.g., conjunctive queries over relational data [TCP14; CD15; BCS16; BR17], SPARQL queries over RDF data [ADK16], and path queries over graph databases [BCL15].

The query-by-example problem in the presence of ontologies is closely related to the area of learning DL concepts from examples, [LH10; Lis12; Tra+14; Fan+18; Fun+19;

Fun19] and to the problem of finding least common subsumers [BST07; Col+16]. A related question is whether two given knowledge bases or two given TBoxes can be distinguished by an OMQ. This question has been studied for $\mathcal{ALC}$ ontologies [Bot+19]. A different learning setting, where a learner can pose queries to an omniscient teacher with the goal to learn a regular language or certain kinds of formulas in propositional logic [Ang87] or, in the field of description logics, an ontology [Kon+17] has been considered as well.

The goal of this chapter is two-fold. First, we aim at initiating research on the QBE approach to querying in the context of ontology-enriched systems. We mainly focus on establishing foundational results for QBE over OES with the ontology formulated in Horn DLs. Formally, we introduce and study the following problem $\textsc{qbe}(\mathcal{L}, \mathcal{Q})$ for an ontology language $\mathcal{L}$ and some query language $\mathcal{Q}$: given an $\mathcal{L}$-KB and sets of positive and negative examples of query answers, decide whether there is a query $q \in \mathcal{Q}$ such that all positive examples are certain answers to q over $\mathcal{K}$, and none of the negative is. As query language $\mathcal{Q}$, we consider CQs and UCQs. We also consider the case where a signature Σ is given and the query has to be formulated over Σ, which is a common feature in many OES. As a simple example, consider the knowledge base consisting of

$$\begin{aligned}\mathcal{T} &= \{\mathsf{Human} \sqsubseteq \mathsf{Vertebrate}, \mathsf{Vertebrate} \sqsubseteq \exists \mathsf{hasPart}.\mathsf{Spine}\},\\ \mathcal{A} &= \{\mathsf{Human}(ax), \mathsf{hasPart}(an, sp), \mathsf{Spine}(sp), \mathsf{Bug}(bug)\}.\end{aligned}$$

If the positive examples are ax, an and the negative example is bug, then

$$q(x) \leftarrow \mathsf{hasPart}(x, y) \wedge \mathsf{Spine}(y)$$

is a witnessing CQ. However, there is no witnessing CQ for the positive examples an, bug if ax is to be avoided.

The second aim is to continue bridging the gap between DL and machine learning research. Indeed, QBE over knowledge bases can be viewed as an instantiation of the *inductive logic programming (ILP)* framework [NW97; Kie02], where background knowledge rules should be learned by observing the data.

The main contributions of this chapter are model-theoretic characterizations, algorithms, and complexity bounds for $\textsc{qbe}(\mathcal{L}, \mathcal{Q})$ for $\mathcal{L} \in \{\mathcal{ELI}, \text{Horn-}\mathcal{ALC}\}$ and $\mathcal{Q} \in \{\text{CQ}, \text{UCQ}\}$. In Section 4.2, we start with providing model-theoretic characterizations for $\textsc{qbe}(\text{Horn-}\mathcal{ALCI}, \mathcal{Q})$ for $\mathcal{Q} \in \{\text{CQ}, \text{UCQ}\}$ by lifting characterizations known from the relational database setting [CD15] by replacing the database with the universal model of the knowledge base. Unfortunately, the characterizations do not give immediate rise to a decision procedure because the universal model is typically infinite. In Section 4.3, we exploit the fact that roles in the anonymous parts of universal models for Horn-$\mathcal{ALC}$ are always directed into the same direction and prove coNExpTime-completeness for Horn-$\mathcal{ALC}$. We then proceed by proving the surprising result that adding inverse roles leads to undecidability, that is $\textsc{qbe}(\mathcal{ELI}, \text{CQ})$ and $\textsc{qbe}(\text{Horn-}\mathcal{ALCI}, \text{CQ})$ are undecidable.

We obtain the same results for the variant $\textsc{qdef}$ of $\textsc{qbe}$, the problem to decide whether some $q \in \mathcal{Q}$ returns *precisely* the positive examples (no negative examples are given).

In Section 4.4, we investigate the *size of witness queries* for the decidable case, Horn-$\mathcal{ALC}$. This is of course vital for practical purposes since at the end the user is interested in obtaining a (witness) query to further explore the data. In particular we show that if a witness query exists, then there is always a witness query of at most double exponential size, and that there are cases where this size is unavoidable. This result is in contrast to the database case, where witness queries only become exponentially large in the worst case.

In Section 4.5, we discuss related work and lay out directions for future work.

4.1 Problem Definition and Basic Observations

We study the following decision problem *Query-by-Example* for some ontology language $\mathcal{L}$ and query language $\mathcal{Q}$:

$\textsc{qbe}(\mathcal{L}, \mathcal{Q})$

Input: A tuple $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$, where $(\mathcal{T}, \mathcal{A})$ is an $\mathcal{L}$-KB, S^+ and S^- are n-ary relations over $\mathsf{ind}(\mathcal{A})$ for some $n \geq 1$ and Σ a signature

Question: Is there a query $q(\mathbf{x}) \in \mathcal{Q}_\Sigma$ such that
- $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ for all $\mathbf{a} \in S^+$, and
- $\mathcal{T}, \mathcal{A} \not\models q(\mathbf{b})$, for all $\mathbf{b} \in S^-$?

We call S^+ and S^- the *positive examples* and the *negative examples*, respectively. A closely related problem is the *query definability problem* $\textsc{qdef}(\mathcal{L}, \mathcal{Q})$, which takes as input a tuple $(\mathcal{T}, \mathcal{A}, S^+, \Sigma)$ and asks whether there is a query $q(\mathbf{x}) \in \mathcal{Q}_\Sigma$ such $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ if and only if $\mathbf{a} \in S^+$. If a tuple $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$ is a yes-instance of $\textsc{qbe}(\mathcal{L}, \mathcal{Q})$, then we call the query $q(\mathbf{x})$ a *witness*. We further define the variant $\textsc{qbe}_f(\mathcal{L}, \mathcal{Q})$ (f standing for *full signature*) as the problem of deciding for a given tuple $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ whether $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma^*) \in \textsc{qbe}(\mathcal{L}, \mathcal{Q})$, where Σ^* is the set of *all* concept and role names occurring in $(\mathcal{T}, \mathcal{A})$. The problem $\textsc{qdef}_f(\mathcal{L}, \mathcal{Q})$ receives only a triple $(\mathcal{T}, \mathcal{A}, S^+)$ and is defined analogously. We always assume $\mathcal{T}$ to be in normal form and it is easy to verify that if $\mathcal{T}'$ is the normalization of $\mathcal{T}$, then $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in \textsc{qbe}(\mathcal{L}, \mathcal{Q})$ if and only if $(\mathcal{T}', \mathcal{A}, S^+, S^-, \Sigma) \in \textsc{qbe}(\mathcal{L}, \mathcal{Q})$, for $\mathcal{Q} \in \{\text{CQ}, \text{UCQ}\}$ and $\mathcal{L} \in \{\mathcal{ELI}, \text{Horn-}\mathcal{ALC}, \text{Horn-}\mathcal{ALCI}\}$ and the same holds for the variant $\textsc{qdef}$ and the variants with full signature. Besides the decision problems, we will also be interested in the size of witness queries (if they exist). Table 4.1 summarizes the results on the decision problems.

From the practical application point of view, the CQ case is arguably more interesting than the UCQ case, since a CQ separating the positive from the negative examples is a generalization of the positive examples, it points out a property that all the positive examples have in common. A witness UCQ, however, can use different disjuncts for the different positive examples, which then do not need to have anything in common. This observation coincides with the picture seen in Table 4.1, that the UCQ case is easier in general. This is illustrated in the following example.

$\mathcal{L} \rightarrow$	$\mathcal{ELI}$ and Horn-$\mathcal{ALCI}$	Horn-$\mathcal{ALC}$
QBE($\mathcal{L}$, CQ)	undecidable	coNExpTime-comp.
QBE$_f$($\mathcal{L}$, CQ)	undecidable	coNExpTime-comp.
QBE($\mathcal{L}$, UCQ)	2-ExpTime-comp. [GJS18b]	ExpTime-comp.
QBE$_f$($\mathcal{L}$, UCQ)	ExpTime-comp. [GJS18b]	ExpTime-comp.

Table 4.1: The table gives an overview over the complexity results on QBE obtained in this chapter. The complexity of QDEF is the same as the complexity of QBE in all these cases.

Example 4.1. *Consider the example knowledge base from the introduction with*

$$\begin{aligned}\mathcal{T} = \{&\mathsf{AlzheimerDisease} \sqsubseteq \mathsf{DementiaDisorder},\\ &\mathsf{DementiaDisorder} \sqsubseteq \exists\, \mathsf{hasFindingSite}.\mathsf{BrainPart},\\ &\mathsf{BrainConcussion} \sqsubseteq \exists\, \mathsf{hasFindingSite}.\mathsf{BrainPart}\}\\ \mathcal{A} = \{&\mathsf{hasFinding}(\mathsf{patient12}, \mathsf{finding345}),\\ &\mathsf{hasFinding}(\mathsf{patient45}, \mathsf{finding257}),\\ &\mathsf{AlzheimerDisease}(\mathsf{finding345}),\\ &\mathsf{BrainConcussion}(\mathsf{finding257})\}\end{aligned}$$

and let $S^+ = \{\mathsf{patient12}, \mathsf{patient45}\}$ *with no negative examples. A witness UCQ is:*

$$q(x) \leftarrow \mathsf{hasFinding}(x, y) \wedge \mathsf{AlzheimerDisease}(y) \vee \mathsf{hasFinding}(x, y) \wedge \mathsf{BrainConcussion}(y)$$

Arguably more interesting, however, is the following CQ, which reveals something that both patients have in common:

$$q(x) \leftarrow \mathsf{hasFinding}(x, y) \wedge \mathsf{hasFindingSite}(y, z) \wedge \mathsf{BrainPart}(z)$$

We deal with two special cases of the QBE problem up front: We always assume that the input knowledge base $(\mathcal{T}, \mathcal{A})$ is consistent and that S^+ is not empty. Both conditions can be effectively checked and if one of them isn't satisfied the reasoning problems become easier. We justify these assumptions in detail.

KB Consistency. We consider first the case where $(\mathcal{T}, \mathcal{A})$ happens to be inconsistent. In that case, we have $\mathcal{T}, \mathcal{A} \models q(\mathbf{a})$ for every n-ary CQ $q(\mathbf{x})$ and every $\mathbf{a} \in \mathsf{ind}(\mathcal{A})^n$, thus there is a witness if and only if $S^- = \emptyset$ (and then, every n-ary CQ is a witness). Hence, one could check $(\mathcal{T}, \mathcal{A})$ for inconsistency first, which can be done in ExpTime if $\mathcal{T}$ is formulated in Horn-$\mathcal{ALCI}$ [KRH13].

No Positive Examples. The second case we discuss is $S^+ = \emptyset$, that is, the question whether there is a $\mathcal{Q}_\Sigma$-query $q(\mathbf{x})$ with n answer variables such that $(\mathcal{T}, \mathcal{A}) \not\models q(\mathbf{b})$ for all

$\mathbf{b} \in S^-$. A natural candidate for such a query is

$$q(x,\dots,x) \leftarrow \bigwedge_{A\in\Sigma} A(x) \wedge \bigwedge_{r\in\Sigma} r(x,x) .$$

It is easy to see that $q(x,\dots,x)$ is the most restrictive query in the sense that if $\mathcal{T},\mathcal{A} \models q(\mathbf{a})$ then $\mathcal{T},\mathcal{A} \models q'(\mathbf{a})$ for every n-ary query $q'(\mathbf{x})$, so the instance has a witness if and only if $q(\mathbf{x})$ is such a witness. Hence, an algorithm for deciding QBE in this special case has to check whether $(\mathcal{T},\mathcal{A}) \not\models q(\mathbf{b})$ for all $\mathbf{b} \in S^-$. For Horn-$\mathcal{ALCI}$ TBoxes, this can be done in ExpTime [KRH13].

Variant with constants. We remark that allowing individual names in witness queries might be desirable in some applications, where the user knows some 'special' individuals which are relevant for her query. Formally this means that a CQ is allowed to contain individual names instead of variables both in the body and in the head. A homomorphism h from a CQ into a model is only valid if $h(a) = a$ for all individual names a that appear in the CQ.

Example 4.2. *Let $\mathcal{T} = \emptyset$, $\mathcal{A} = \{r(a,b), r(c,d)\}$, $S^- = \{(a,b)\}$, $S^- = \{(c,d)\}$ and $\Sigma = \{A, r\}$. Without using individual names in the query, the pairs (a,b) and (c,d) are obviously indistinguishable, so this is a no-instance of* QBE. *However, if we allow the individual name a, then $q(a,y) \leftarrow r(a,y)$ is a witness. If we allow the individual name b, then $q(x,b) \leftarrow r(x,b)$ is a witness.*

Let $\text{QBE}^c(\mathcal{L},\mathcal{Q})$ be the generalization of $\text{QBE}(\mathcal{L},\mathcal{Q})$ that takes another input $I \subseteq \text{ind}(\mathcal{A})$ and allows the witness query to use individual names from I. We then have:

Lemma 4.3. *$\text{QBE}^c(\mathcal{L},\mathcal{Q})$ and $\text{QDEF}^c(\mathcal{L},\mathcal{Q})$ reduce in polynomial time to $\text{QBE}(\mathcal{L},\mathcal{Q})$ and $\text{QDEF}(\mathcal{L},\mathcal{Q})$, respectively, for $\mathcal{Q} \in \{CQ, UCQ\}$, for any $\mathcal{L}$.*

Proof. We prove it for $\mathcal{Q} = \text{CQ}$, for UCQs it is similar. Let $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma,I)$ be an instance of $\text{QBE}^c(\mathcal{L},\mathcal{Q})$. Define $\Sigma' = \Sigma \cup \{X_a \mid a \in I\}$, where all X_a are fresh concept names, and $\mathcal{A}' = \mathcal{A} \cup \{X_a(a) \mid a \in I\}$. It is routine to verify correctness of the reduction:

Claim. $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma,I) \in \text{QBE}_c(\mathcal{L},\mathcal{Q})$ if and only if $(\mathcal{T},\mathcal{A}',S^+,S^-,\Sigma') \in \text{QBE}(\mathcal{L},\mathcal{Q})$.

For the 'only if' direction, let $q(\mathbf{x})$ be a witness for $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma,I) \in \text{QBE}^c(\mathcal{L},\mathcal{Q})$. Obtain a query $q' \in \mathcal{L}_{\Sigma'}$ from q as follows: For every $a \in I$, replace all occurrences of a in body and head with a fresh variable x_a and add the conjunct $X_a(x_a)$ to the body.

It should be clear that $\mathcal{T},\mathcal{A}' \models q'(\mathbf{a})$ for all $\mathbf{a} \in S^+$. Assume now that $\mathcal{T},\mathcal{A}' \models q'(\mathbf{a})$ for some $\mathbf{a} \in S^-$, and let $\mathcal{I}$ be an arbitrary model of $(\mathcal{T},\mathcal{A})$. Obviously, the extension $\mathcal{I}'$ of $\mathcal{I}$ interpreting every fresh concept X_a with $X_a^{\mathcal{I}'} = \{a\}$ is a model of $(\mathcal{T},\mathcal{A}')$. Let h be a homomorphism from $q'(\mathbf{a})$ to $\mathcal{I}'$. By construction, h is also a homomorphism from $q(\mathbf{a})$ into $\mathcal{I}$. Hence, we obtain $\mathcal{T},\mathcal{A} \models q(\mathbf{a})$, a contradiction.

For the 'if' direction, let $q(\mathbf{x})$ be a witness for $(\mathcal{T},\mathcal{A}',S^+,S^-,\Sigma') \in \text{QBE}(\mathcal{L},\mathcal{Q})$. Note that it cannot be the case that there is a variable z such that both $X_a(z)$ and $X_b(z)$ for $a \neq b$ appear in the body of q, since this implies $S^+ = \emptyset$, contradicting our assumption about S^+. Obtain a query q' from q as follows: For every variable z such that $X_a(z)$ appears in the body of q, replace all occurrences of z in the body or the head of q with a. It is routine to verify that $q'(\mathbf{x})$ witnesses that $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma,I) \in \text{QBE}_c(\mathcal{L})$. □

4.2 Model-Theoretic Characterizations

In this section, we provide model-theoretic characterizations of QBE and QDEF, setting the foundations for the development of the decision procedures later on. The characterization is based on the notion of direct products. Let $\mathcal{I}_1, \dots, \mathcal{I}_n$ be interpretations. The *direct product* $\prod_{i=1}^n \mathcal{I}_i$ is the interpretation defined by

$$\begin{aligned}
\Delta^{\prod_{i=1}^n \mathcal{I}_i} &= \Delta^{\mathcal{I}_1} \times \dots \times \Delta^{\mathcal{I}_n},\\
A^{\prod_{i=1}^n \mathcal{I}_i} &= A^{\mathcal{I}_1} \times \dots \times A^{\mathcal{I}_n},\\
r^{\prod_{i=1}^n \mathcal{I}_i} &= \{((a_1, \dots, a_n), (b_1, \dots, b_n)) \mid (a_i, b_i) \in r^{\mathcal{I}_i} \text{ for all } i \in \{1, \dots, n\}\},
\end{aligned}$$

for all concept names A and role names r. For interpretations with a distinguished tuple $(\mathcal{I}_1, \mathbf{a}_1), \dots, (\mathcal{I}_n, \mathbf{a}_n)$, where all $\mathbf{a}_i$ have length k, the direct product is again an interpretation with a distinguished tuple of length k, defined by $\prod_{i=1}^n(\mathcal{I}_i, \mathbf{a}_i) = (\prod_{i=1}^n \mathcal{I}_i, \prod_{i=1}^n \mathbf{a}_i)$, where $\prod_{i=1}^n \mathbf{a}_i = ((a_1^1, \dots, a_n^1), \dots, (a_1^k, \dots, a_n^k))$, assuming $\mathbf{a}_i = (a_i^1, \dots, a_i^k)$. For a direct product of only two interpretations (with distinguished tuple), we write $(\mathcal{I}_1, \mathbf{a}_1) \otimes (\mathcal{I}_2, \mathbf{a}_2)$ instead of $\prod_{i=1}^2(\mathcal{I}_i, \mathbf{a}_i)$. Given Σ, a product $\prod_{i=1}^n(\mathcal{I}_i, \mathbf{a}_i)$ is called Σ-*safe* if every element of the tuple $\prod_{i=1}^n \mathbf{a}_i$ appears in the extension of some concept or role name from Σ in $\prod_{i=1}^n(\mathcal{I}_i, \mathbf{a}_i)$; again, we drop Σ in case it is trivial. Note that this definition of safety is consistent with the safety of CQs, defined in Section 2.4, when the the CQ is seen as an interpretation with the tuple of answer variables being the distinguished tuple.

Let us recall the characterization for QBE with CQs over relational databases [CD15; BR17]. For the sake of simplicity, we state it here in our terminology, that is, consider ABoxes instead of databases. Given an ABox $\mathcal{A}$ and sets S^+, S^- of examples over $\mathcal{A}$, there is a CQ distinguishing S^+ and S^- if and only if

1. $\prod_{\mathbf{a} \in S^+}(\mathcal{I}_\mathcal{A}, \mathbf{a})$ is safe, and
2. $\prod_{\mathbf{a} \in S^+}(\mathcal{I}_\mathcal{A}, \mathbf{a}) \not\to (\mathcal{I}_\mathcal{A}, \mathbf{b})$ for every $\mathbf{b} \in S^-$,

where $\mathcal{I}_\mathcal{A}$ is $\mathcal{A}$ viewed as an interpretation. The intuition behind this characterization is as follows: the constructed product can be viewed as CQ with answer variables $\prod_{\mathbf{a} \in S^+} \mathbf{a}$; in fact, this CQ is the *least general generalization*, a well known concept in machine learning [Plo70], of the positive examples. Condition 1 ensures that this CQ is safe and Condition 2 ensures that no negative examples are returned.

We argue, however, that this simple characterization does not apply to the case with ontologies. In fact, the example from the introduction to this chapter does not satisfy Condition 1, but there exists a witness query. We lift the characterization to take into account non-empty TBoxes using universal models, introduced in Section 2.6. We start with the characterization for $\mathcal{Q} = \text{CQ}$.

Theorem 4.4. *For every Horn-$\mathcal{ALCI}$-KB $(\mathcal{T}, \mathcal{A})$, all n-ary relations S^+ and S^- over* $\mathsf{ind}(\mathcal{A})$*, and signature Σ, we have:*

- $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in$ QBE*(Horn-$\mathcal{ALCI}$, CQ) if and only if*

  1. $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a})$ *is* Σ*-safe, and*
  2. $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a}) \not\rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{b})$ *for all* $\mathbf{b} \in S^-$.

- $(\mathcal{T},\mathcal{A},S^+,\Sigma) \in$ *QDEF(Horn-*$\mathcal{ALCI}$*, CQ) if and only if*
  1. $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a})$ *is* Σ*-safe, and*
  2. $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a}) \not\rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{b})$ *for all* $\mathbf{b} \in \mathsf{ind}(\mathcal{A})^n \setminus S^+$.

Proof. The characterization for QDEF(Horn-$\mathcal{ALCI}$, CQ) follows immediately from the characterization for QBE(Horn-$\mathcal{ALCI}$, CQ), so we only need to prove the characterization for QBE(Horn-$\mathcal{ALCI}$, CQ).

($\Rightarrow$) Let $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma) \in$ QBE(Horn-$\mathcal{ALCI}$, CQ) with $S^+ = \{\mathbf{a}_1,\ldots,\mathbf{a}_m\}$ and let $q(\mathbf{x})$ be a Σ-CQ witnessing this. By Lemma 2.1, for every i, there is a homomorphism h_i from $q(\mathbf{x})$ into $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h_i(\mathbf{x}) = \mathbf{a}_i$. Define h by taking $h(z) = (h_1(z),\ldots,h_m(z))$, for every variable z in $q(\mathbf{x})$. By construction, h is a homomorphism from $q(\mathbf{x})$ to $\prod_{\mathbf{a}\in S^+}\mathcal{U}_{\mathcal{A},\mathcal{T}}$ and $h(\mathbf{x}) = \mathbf{a}_1\otimes\ldots\otimes\mathbf{a}_m$, which is thus Σ-safe. Assume that Condition 2 does not hold, that is, there is a $\mathbf{b} \in S^-$ such that there is a homomorphism $g : \prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a}) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{b})$. Then the composition $h' = g \circ h$ is a homomorphism from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h'(\mathbf{x}) = \mathbf{b}$. Hence, $\mathcal{T},\mathcal{A} \models q(\mathbf{b})$, a contradiction.

($\Leftarrow$) For the other direction, let Conditions 1 and 2 be fulfilled. We show that there is a witness for $(\mathcal{T},\mathcal{A},S^+,S^-,\Sigma) \in$ QBE(Horn-$\mathcal{ALCI}$, CQ). Let $(\mathcal{I},\mathbf{a}^*)$ be the Σ-restriction of $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a})$, and let $q(\mathbf{x})$ be $(\mathcal{I},\mathbf{a}^*)$ viewed as a (possibly infinite) CQ; in particular, $\mathbf{a}^*$ becomes the tuple of answer variables $\mathbf{x}$. By Condition 1, $q(\mathbf{x})$ has the right number of answer variables. Clearly, every $\mathbf{a} \in S^+$ is a certain answer to $q(\mathbf{x})$, using the projection mappings, and none of the $\mathbf{b} \in S^-$ is a certain answer by Condition 2 and universality of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. If $q(\mathbf{x})$ is finite, we are done. If $q(\mathbf{x})$ is infinite, we show that there is a finite subquery of $q(\mathbf{x})$ which is a witness. Denote with $q_i(\mathbf{x})$, $i \geq 0$, the restriction of $q(\mathbf{x})$ to variables that have distance at most i to the answer variables $\mathbf{x}$ in $q(\mathbf{x})$. Since $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ has finite degree, so has $\prod_{\mathbf{a}\in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}},\mathbf{a})$ and also $q(\mathbf{x})$. Thus, $q_i(\mathbf{x})$ is finite for every $i \geq 1$. Since every q_i is a restriction of q and since $\mathcal{T},\mathcal{A} \models q(\mathbf{a})$ for all $\mathbf{a} \in S^+$, we also have $\mathcal{T},\mathcal{A} \models q_i(\mathbf{a})$ for all $\mathbf{a} \in S^+$ and all i.

We show that one of the q_i must be a witness. To obtain a contradiction, assume that q_i is not a witness for every $i \geq 1$, that is, there are homomorphisms h_i from $q_i(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h_i(\mathbf{x}) = \mathbf{b}_i$ for some $\mathbf{b}_i \in S^-$. Since S^- is finite, there is some $\mathbf{b} \in S^-$ such that $\mathbf{b} = \mathbf{b}_i$ for infinitely many i. Thus, there are homomorphisms h_i, $i \geq 1$, from $q_i(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h_i(\mathbf{x}) = \mathbf{b}$. We construct a sequence of homomorphisms $(h'_i)_{i\geq 1}$ such that for all $j \geq 1$, we have

($*$) for all $k \in \{1,\ldots,j-1\}$, $h'_k(z) = h'_j(z)$ for all $z \in \mathsf{var}(q_i)$.

Start with setting $h'_1 = h_1$, obviously satisfying ($*$). To define h'_j, assume that h'_k are defined for all $k \in \{1,\ldots,j-1\}$. Let $V_j = \mathsf{var}(q_j) \setminus \mathsf{var}(q_{j-1})$, and define, for all $k \geq j$, g_k as the restriction of h_k to V_j. By construction V_j is finite. Moreover, as $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ has finite outdegree, there are only finitely many different g_k. Choose some g such that $g = g_k$ for infinitely many $k \geq j$. Then obtain a new sequence of homomorphisms by dropping all h_k such that $g_k \neq g$. Setting $h'_j = g \cup h'_{j-1}$ finishes the construction and satisfies ($*$).

It remains to note that $\hat{h} = \bigcup_{i \geq 0} h'_i$ is a homomorphism from $q(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $\hat{h}(\mathbf{x}) = \mathbf{b}$. Thus, $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, contradicting Condition 2. □

Theorem 4.4 shows that the characterization is the same as in the database setting, but with $\mathcal{I}_\mathcal{A}$ replaced by $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. Note that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is possibly infinite, so the product is, in contrast to the database case, *not* the witness. In fact, the proof for direction ($\Leftarrow$) merely shows that *there is* a witness, but in a non-constructive way. Hence, Theorem 4.4 does not give immediate bounds on the size of witness queries and does not immediately yield a decision procedure.

In case of UCQs the additional expressive power leaves us with a simpler characterization, the product in each second point is compensated for by the use of disjunction in the query language and is thus not necessary anymore.

Theorem 4.5. *For every Horn-$\mathcal{ALCI}$-KB $(\mathcal{T}, \mathcal{A})$, all n-ary relations S^+ and S^- over* $\mathsf{ind}(\mathcal{A})$*, and signatures Σ, we have:*

- $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in$ QBE*(Horn-$\mathcal{ALCI}$, UCQ) if and only if*
  1. $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ *is Σ-safe and*
  2. $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \not\rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$*, for all $\mathbf{a} \in S^+$ and $\mathbf{b} \in S^-$.*
- $(\mathcal{T}, \mathcal{A}, S^+, \Sigma) \in$ QDEF*(Horn-$\mathcal{ALCI}$, UCQ) if and only if*
  1. $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ *is Σ-safe and*
  2. $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \not\rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ *for all $\mathbf{a} \in S^+$ and $\mathbf{b} \in \mathsf{ind}(\mathcal{A})^n \setminus S^+$.*

Proof. ($\Rightarrow$) Let $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in$ QBE(Horn-$\mathcal{ALCI}$, UCQ), witnessed by a Σ-UCQ $q(\mathbf{x})$. By universality, for every $\mathbf{a} \in S^+$, there is a disjunct $q'(\mathbf{x})$ of $q(\mathbf{x})$ such that there is a homomorphism h from $q'(\mathbf{x})$ to $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$. Since q' is a Σ-query, $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ is Σ-safe. Suppose now that there are $\mathbf{a} \in S^+$, $\mathbf{b} \in S^-$ such that there is a Σ-homomorphism $g : (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$. As $\mathbf{a} \in S^+$ and $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is universal, there is a homomorphism h from a disjunct $q'(\mathbf{x})$ of q into $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ with $h(\mathbf{x}) = \mathbf{a}$. Composing g and h yields a homomorphism from $q'(\mathbf{x})$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, thus $\mathcal{T}, \mathcal{A} \models q(\mathbf{b})$, a contradiction to $\mathbf{b} \in S^-$.

($\Leftarrow$) For the other direction, let $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ be Σ-safe and $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \not\rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, for all $\mathbf{a} \in S^+$ and $\mathbf{b} \in S^-$. We show that there is witness for $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in$ QBE(Horn-$\mathcal{ALCI}$, UCQ).

For the sake of simplicity, we abbreviate $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ just with $\mathcal{I}$. Further, we denote with $q_{\mathcal{I},\mathbf{a}}(\mathbf{x})$ the interpretation $(\mathcal{I}, \mathbf{a})$ viewed as (possibly infinite) CQ with the answer variables $\mathbf{x}$ being the distinguished tuple $\mathbf{a}$. For a (U)CQ q, denote with $q^\Sigma(\mathbf{x})$ the restriction of $q(\mathbf{x})$ to symbols from Σ, and with $q^i(\mathbf{x})$ the UCQ obtained from $q(\mathbf{x})$ by restricting every disjunct to variables that have distance at most i to the answer variables of that disjunct.

We now define a UCQ q where the disjuncts can possibly be of infinite size: Let $q = \bigvee_{\mathbf{a} \in S^+} q^\Sigma_{\mathcal{I},\mathbf{a}}$. By Σ-safety and universality of $\mathcal{I}$, every $\mathbf{a} \in S^+$ is a certain answer to this query and none of the $\mathbf{b} \in S^-$ is a certain answer. Thus, if all disjuncts of q are finite, then q is the required witness. If q this is not the case, we show we can restrict every disjunct to a finite subset of to obtain a witness for $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$. Assume the opposite, that is, $q^i(\mathbf{x})$ is not a witness for every $i \geq 1$. To reach a contradiction, we show that there exist

$\mathbf{a} \in S^+$, $\mathbf{b} \in S^-$ with $(\mathcal{I}, \mathbf{a}) \rightarrow_\Sigma (\mathcal{I}, \mathbf{b})$, contradicting Condition 1. Clearly, for every such $q^i(\mathbf{x})$ we still have $\mathcal{T}, \mathcal{A} \models q^i(\mathbf{a})$ for all $\mathbf{a} \in S^+$. However, by our assumption, for every i, there is a $\mathbf{b}_i \in S^-$ such that $\mathcal{T}, \mathcal{A} \models q^i(\mathbf{b}_i)$. Since S^- is finite and $q(\mathbf{x})$ consists of finitely many disjuncts, there have to be $\mathbf{a} \in S^+$ and $\mathbf{b} \in S^-$ such that, for infinitely many i:

- $\mathbf{b} = \mathbf{b}_i$, and
- the disjunct $p_i = q^{\Sigma,i}_{\mathcal{I},\mathbf{a}}$ of q_i corresponding to $\mathbf{a}$, satisfies $\mathcal{T}, \mathcal{A} \models p_i(\mathbf{b}_i)$.

We can then proceed as in the proof of Theorem 4.4 and construct a homomorphism $(\mathcal{I}, \mathbf{a}) \rightarrow_\Sigma (\mathcal{I}, \mathbf{b})$. □

4.3 Complexity of QBE and QDEF

Based on the characterizations in Theorems 4.4 and 4.5, we now pinpoint the complexity for the introduced decision problems. We start with observing that Σ-safety (in both theorems) can be checked in exponential time by computing first $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ up to depth 1, computing the product (only in case of Theorem 4.4), and directly checking the condition.

Lemma 4.6. *Given $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$ with $\mathcal{T}$ formulated in Horn-$\mathcal{ALCI}$, it is ExpTime-complete to decide whether $\prod_{\mathbf{a} \in S^+} \mathcal{U}_{\mathcal{A},\mathcal{T}}$ is Σ-safe.*

Proof. It suffices to compute $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ restricted to elements of $\prod_{\mathbf{a} \in S^+} \mathbf{a}$ and all neighbours of such elements. This can be done by computing the universal model of $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ up to depth one, which can be done in ExpTime [KRH13].

For the lower bound, we reduce from the subsumption problem in $\mathcal{ELI}$, which is ExpTime-hard [BLB08]. Let $A_1, A_2 \in \mathsf{N_C}$ be concept names and $\mathcal{T}$ an $\mathcal{ELI}$-TBox. The question is whether $\mathcal{T} \models A_1 \sqsubseteq A_2$. We construct an instance $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$, where $\mathcal{T}$ is the same $\mathcal{ELI}$-TBox, $\mathcal{A} = \{A_1(a)\}$, $S^+ = \{a\}$, $S^- = \emptyset$ and $\Sigma = \{A_2\}$. Now it is clear that $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ is Σ-safe if and only if $\mathcal{T} \models A_1 \sqsubseteq A_2$. □

Checking the other conditions of Theorems 4.4 and 4.5 comes down to finding an algorithm for deciding the following for a given $\mathbf{b} \in S^-$:

$$\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b}) \tag{4.1}$$

An algorithm deciding (4.1) can also be used for the homomorphism checks in Theorem 4.5 by treating the elements $\mathbf{a} \in S^+$ individually. By Lemma 4.6, safety can be decided in ExpTime, but as Table 4.1 shows, the complexity of QBE is usually higher than ExpTime, so Table 4.1 shows the complexity of deciding (4.1), which varies for the different variations of the QBE problem.

If $\mathcal{T}$ is formulated in Horn-$\mathcal{ALC}$, then $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ is a pseudo tree-shaped structure with core $\prod_{\mathbf{a} \in S^+} \mathsf{ind}(\mathcal{A})$ and we can find coNExpTime or ExpTime algorithms for deciding (4.1). For QBE($\mathcal{ELI}$, CQ) however, the problem becomes undecidable. Intuitively, this is due to the fact that the product of universal models with inverse roles can from complex structures that are not pseudo tree-shaped any more. For the variations QBE(Horn-$\mathcal{ALCI}$, UCQ) and QBE$_f$(Horn-$\mathcal{ALCI}$, UCQ), we do not have to construct a product, which is why these variations are decidable again.

Remark 4.7. *In the conference paper [GJS18b], the paper on which this chapter is based, it was claimed that* QBE(Horn-$\mathcal{ALCI}$, CQ) *and* QBE$_f$(Horn-$\mathcal{ALCI}$, CQ) *are decidable and 2-ExpTime-complete. This result is incorrect. The mistake was caused by our false assumption that even for Horn-$\mathcal{ALCI}$ TBoxes, the anonymous parts of the product of universal models are still regular trees.*

4.3.1 Horn-$\mathcal{ALC}$

Let us fix an input $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$, where $\mathcal{T}$ is formulated in Horn-$\mathcal{ALC}$. We develop a decision procedure for (4.1). Let $k = |S^+|$ and denote with $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$ the product $\prod_{i=1}^k \mathcal{U}_{\mathcal{A},\mathcal{T}}$. Observe that $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$ is pseudo tree-shaped with core $\mathsf{ind}(\mathcal{A})^k$, which is due to the fact that edges in the anonymous part of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ are always directed away from the core when $\mathcal{T}$ is formulated in Horn-$\mathcal{ALC}$. The product $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$ might be disconnected and for our purposes it suffices to consider the substructure $\mathcal{P}$ of $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$ containing all elements from $\mathsf{ind}(\mathcal{A})^k$ and everything that is reachable from there; thus, the domain $\Delta^{\mathcal{P}}$ of $\mathcal{P}$ is the smallest set such that:

- $\mathsf{ind}(\mathcal{A})^k \subseteq \Delta^{\mathcal{P}}$ and
- whenever $\mathbf{p} \in \Delta^{\mathcal{P}}$ and $(\mathbf{p}, \mathbf{p}') \in r^{\mathcal{U}^k_{\mathcal{A},\mathcal{T}}}$, then also $\mathbf{p}' \in \Delta^{\mathcal{P}}$

It is easy to show that for $\mathbf{a}^* = \prod_{\mathbf{a} \in S^+} \mathbf{a}$, we have:

Lemma 4.8. *Let $\mathbf{b} \in S^-$. Then $(\mathcal{U}^k_{\mathcal{A},\mathcal{T}}, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ if and only if $(\mathcal{P}, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$.*

Proof. The direction ($\Rightarrow$) is trivial since $\mathcal{P}$ is a sub-interpretation of $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$.

For ($\Leftarrow$), let $h : (\mathcal{P}, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$. Note that, by definition, $\mathcal{P}$ is the union of all maximal connected sub-interpretations containing individuals from $\mathsf{ind}(\mathcal{A})^k$. We can extend the homomorphism h to the remaining connected components $\mathcal{I} \neq \mathcal{P}$ of $\mathcal{U}^k_{\mathcal{A},\mathcal{T}}$ by taking the projection of $\Delta^{\mathcal{I}}$ to an arbitrary (but fixed) component. □

For what follows, it is convenient to characterize $r^{\mathcal{P}}$ in terms of (tuples of) *types*, similar to the definition of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. For doing so, let TP be the set of all $\mathcal{T}$-types and $\Delta = \mathsf{ind}(\mathcal{A}) \cup \mathsf{TP}$. Then define, for each role r, a binary relation $\hookrightarrow_r^{\mathcal{T},\mathcal{A}}$ on Δ^k by taking $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$ if and only if $\mathbf{c} = (c_1, \dots, c_k)$ and $\mathbf{d} = (d_1, \dots, d_k)$ and for each $1 \leq i \leq k$ we have:

- if $c_i, d_i \in \mathsf{ind}(\mathcal{A})$, then $r(c_i, d_i) \in \mathcal{A}$;
- if $c_i \in \mathsf{ind}(\mathcal{A}), d_i \in \mathsf{TP}$, then $c_i \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} d_i$;
- if $c_i, d_i \in \mathsf{TP}$, then $c_i \rightsquigarrow_r^{\mathcal{T}} d_i$.

For $\mathbf{p} = (\pi_1, \ldots, \pi_k) \in \Delta^{\mathcal{P}}$, denote with $\mathsf{tail}(\mathbf{p})$ the tuple $(\mathsf{tail}(\pi_1), \ldots, \mathsf{tail}(\pi_k))$. It should be clear that we have $(\mathbf{p}, \mathbf{p}') \in r^{\mathcal{P}}$ if and only if $\mathsf{tail}(\mathbf{p}) \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathsf{tail}(\mathbf{p}')$.

We give a characterization for $(\mathcal{P}, \mathbf{a}^*) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, which will be the basis of the decision procedure. Intuitively, we decompose $\mathcal{P}$ into the core part with domain $N = \mathsf{ind}(\mathcal{A})^k$ and the tree-shaped subinterpretations below each $\mathbf{c} \in N$, which are characterized alone by their roots. For $\mathbf{c} \in \Delta^k$, we denote the tree-shaped subinterpretation rooted at $\mathbf{c}$ by $\mathcal{P}_\mathbf{c}$. Moreover, we use the notation $\mathcal{U}_t$ for a $\mathcal{T}$-type t as an abbreviation for the universal model $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, where $\mathcal{A} = \{B(a_t) \mid B \in t\}$ and denote with a_t its root. We define a relation $\mathsf{Hom} \subseteq \Delta^k \times \mathsf{TP}$ that contains the information about which subtrees of $\mathcal{P}_\mathbf{c}$ can be mapped to which subtrees of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$. Given a tuple $\mathbf{c} \in \Delta^k$, and a type $t \in \mathsf{TP}$, we define $(\mathbf{c}, t) \in \mathsf{Hom}$ if there is a homomorphism g from $\mathcal{P}_\mathbf{c}$ to $\mathcal{U}_t$ which maps the root to the root. (Note that both interpretations are tree-shaped.) For $\mathbf{c} \in \Delta^k$, $t \in \mathsf{TP}$ we write $\mathbf{c} \rightarrow_\Sigma t$ if there is a Σ-homomorphism from an element of type $\mathbf{c}$ to an element of type t. We further denote with $\mathsf{tp}_{\mathcal{U}_{\mathcal{A},\mathcal{T}}}(\pi)$ the type of π in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$ and with $\mathcal{P}|_N$ the restriction of $\mathcal{P}$ to domain N. We establish the following characterization.

Lemma 4.9. $(\mathcal{P}, \mathbf{a}^*) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ *if and only if there exists a Σ-homomorphism h : $(\mathcal{P}|_N, \mathbf{a}^*) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ and a labelling $L(\pi) \subseteq \Delta^k$ for every $\pi \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$ such that:*

1. *for every $\mathbf{p} \in N$, we have $\mathbf{p} \in L(h(\mathbf{p}))$;*
2. *for every $\mathbf{c} \in L(\pi)$, we have $\mathbf{c} \rightarrow_\Sigma \mathsf{tp}_{\mathcal{U}_{\mathcal{A},\mathcal{T}}}(\pi)$;*
3. *for every $\pi \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, every $\mathbf{c} \in L(\pi)$, and every $\mathbf{d}$ with $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$ one of the following is true:*
   a) *there is $\pi' \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$ such that $(\pi, \pi') \in r^{\mathcal{U}_{\mathcal{A},\mathcal{T}}}$ and $\mathbf{d} \in L(\pi')$, or*
   b) *$\pi \rightsquigarrow_r^{\mathcal{A},\mathcal{T}} t'$ and $(\mathbf{d}, t') \in \mathsf{Hom}$.*

Proof. For the 'only if' direction, let $g : (\mathcal{P}, \mathbf{a}^*) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$. We define $h = g|_N$ and, for $a \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, we set

$$L(a) = \{\mathsf{tail}(\mathbf{p}) \mid \mathbf{p} \in \Delta^{\mathcal{P}} \wedge g(\mathbf{p}) = a\}.$$

We verify h and T satisfy Conditions 1–3.

Condition 1. Let $\mathbf{p} \in N$. We have $\mathbf{p} \in h^{-1}(h(\mathbf{p})) \subseteq g^{-1}(h(\mathbf{p}))$, which implies $\mathbf{p} \in L(h(\mathbf{p}))$.

Condition 2. Let $\pi \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, $\mathbf{c} \in L(\pi)$ and $A \in \Sigma$. Since $\mathbf{c} \in L(\pi)$, there exists a $\mathbf{p} \in \Delta^{\mathcal{P}}$ such that $g(\mathbf{p}) = \pi$ and $\mathsf{tail}(\mathbf{p}) = \mathbf{c}$. Since $A \in \mathbf{c}$, by the definition of $\mathcal{P}$, we have $\mathbf{p} \in A^{\mathcal{P}}$. Since g is a Σ-homomorphism, $\pi = g(\mathbf{p}) \in A^{\mathcal{U}_\mathcal{K}}$.

Condition 3. Let $\pi \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, $\mathbf{c} \in L(\pi)$ and $\mathbf{d} \in \Delta^k$ with $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$. Since $\mathbf{c} \in L(\pi)$, there exists a $\mathbf{p} \in \Delta^{\mathcal{P}}$ such that $g(\mathbf{p}) = \pi$ and $\mathsf{tail}(\mathbf{p}) = \mathbf{c}$. We distinguish two cases, depending on $b := g(\mathbf{p}r\mathbf{d})$:

- If $b \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, then it is clear that $(a, b) \in r^{\mathcal{U}_\mathcal{K}}$ and $\mathbf{d} \in T(b)$. In this case, Condition 3(a) holds.

- If $b \notin \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$, then b is an anonymous element in $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, introduced via $\pi \rightsquigarrow_r^{\mathcal{A},\mathcal{T}} t'$ for some $t' \in \mathsf{TP}$. Since both $\mathcal{P}_{\mathbf{d}}$ and $\mathcal{U}_{t'}$ have the shaped of a directed tree, g maps the subtree rooted at $\mathbf{d}$ completely into to subtree rooted at b. Thus, $(t', \mathbf{d}) \in \mathsf{Hom}$.

For the 'if' direction, let $h : (\mathcal{P}|_N, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_\mathcal{K}, \mathbf{b})$ and $L : \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A}) \to 2^{\Delta^k}$ such that Conditions 1 to 3 are fulfilled.

For constructing the homomorphism $g : (\mathcal{P}, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_\mathcal{K}, \mathbf{b})$, we construct a series $g_0, g_1, \ldots$ of homomorphisms with increasing domains $N = \mathsf{dom}(g_0) \subseteq \mathsf{dom}(g_1) \subseteq \ldots$, such that every g_{i+1} extends g_i and $\bigcup_{i=0}^\infty \mathsf{dom}(g_i) = \Delta^\mathcal{P}$, and we will then set $g = \bigcup_{i=0}^\infty g_i$.

Set $g_0 = h$. If g_i has been defined and $\mathsf{dom}(g_i) \subsetneq \Delta^\mathcal{P}$, we define g_{i+1} in the following way: Choose a leaf $\mathbf{pc} \in \mathsf{dom}(g_i)$ such that there exists $\mathbf{pc}r\mathbf{d} \notin \mathsf{dom}(g_i)$ for some role r and some $\mathbf{d} \in \Delta^k$. If Condition 3(a) is fulfilled, we set $g_{i+1}(\mathbf{d}) = \pi'$ for the π' guaranteed by that Condition. Otherwise, Condition 3(b) is fulfilled and we define g_{i+1} for the whole subtree rooted at $\mathbf{d}$ according to the homomorphism guaranteed by $(\mathbf{d}, t') \in \mathsf{Hom}$.

If the leafs $\mathbf{pc}$ are chosen in a fair way, i.e. ensuring that every such leaf gets chosen at some point, it follows that $\bigcup_{i=0}^\infty \mathsf{dom}(g_i) = \Delta^\mathcal{P}$, since $\mathcal{P}$ is connected. Since all functions used to construct g are Σ-homomorphisms, g is a Σ-homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_\mathcal{K}, \mathbf{b})$. □

To obtain a decision procedure for QBE(Horn-$\mathcal{ALC}$, CQ) from Lemma 4.9, we need to decide the relation Hom.

Lemma 4.10. *Given $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$, where $\mathcal{T}$ is formulated in Horn-$\mathcal{ALC}$, the problem of deciding* Hom *is in* ExpTime.

Proof. We give an ExpTime procedure for computing the relation Hom. We define a sequence of relations $\mathsf{Hom}_i \subseteq \Delta^k \times \mathsf{TP}$ that approximates Hom from above. Let $\mathbf{c} = (\pi_1, \ldots, \pi_k)$. By $\mathsf{tp}(\pi_i)$ we denote the set of concept names that are true at π_i, i.e. if $\pi_i = a$ for some $a \in \mathsf{ind}(\mathcal{A})$, then $\mathsf{tp}(\pi_i) = \mathsf{tp}_{\mathcal{T},\mathcal{A}}(a)$ and otherwise, $\mathsf{tp}(\pi_i) = \mathsf{tail}(\pi_i)$. Let Hom_0 contain all pairs $(\mathbf{c}, t)$ such that $\bigcap_{i=1}^k \mathsf{tp}(\pi_i) \subseteq t$. A pair $(\mathbf{c}, t)$ is in Hom_{i+1} if it is in Hom_i and for every $\mathbf{d} \in \Delta^k$ and $r \in \Sigma$ with $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$ there exists a type t' such that $t \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} t'$ and $(\mathbf{d}, t') \in \mathsf{Hom}_i$.

Claim: $\mathsf{Hom} = \bigcap_{i=0}^\infty \mathsf{Hom}_i$.

Proof of the Claim. ($\subseteq$): First we show by induction on i:

$$\mathsf{Hom} \cap \mathsf{Hom}_i \subseteq \mathsf{Hom}_{i+1} \tag{*}$$

– We begin by showing $\mathsf{Hom} \subseteq \mathsf{Hom}_0 \cap \mathsf{Hom}_1$, which implies the induction start. Let $(\mathbf{c}, t) \in \mathsf{Hom}$, so there exists $h : (\mathcal{P}_\mathbf{c}, \mathbf{c}) \to_\Sigma (\mathcal{U}_{t(a),\mathcal{T}}, a)$. Since $h(\mathbf{c}) = t$, we have for every concept name $A \in \Sigma$ that $\mathbf{c} \in A^{\mathcal{P}_\mathbf{c}}$ implies $A \in t$ and it follows that $(\mathbf{c}, t) \in \mathsf{Hom}_0$. Let $r \in \Sigma$ and $\mathbf{d} \in \Delta^k$ such that $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$. Since h is defined for $\mathbf{d}$, it follows that there is a $t' \in \mathsf{TP}$ with $t \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} t'$ and it follows that $(\mathbf{d}, t') \in \mathsf{Hom}_0$. Therefore, $(\mathbf{c}, t) \in \mathsf{Hom}_1$.

– For the induction step, assume $\mathsf{Hom} \cap \mathsf{Hom}_i \subseteq \mathsf{Hom}_{i+1}$. We need to show that $\mathsf{Hom} \cap \mathsf{Hom}_{i+1} \subseteq \mathsf{Hom}_{i+2}$. Let $(\mathbf{c}, t) \in \mathsf{Hom} \cap \mathsf{Hom}_{i+1}$. Thus, for every $r \in \Sigma$ and $\mathbf{d} \in \Delta^k$

with $\mathbf{c} \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$, there exists a $t_{\mathbf{d}} \in \mathsf{TP}$ such that $t \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} t_{\mathbf{d}}$ and $(\mathbf{d}, t_{\mathbf{d}}) \in \mathsf{Hom}_i$. But we also have $(\mathcal{P}_{\mathbf{d}}, \mathbf{d}) \rightarrow_\Sigma (\mathcal{U}_{t_{\mathbf{d}}(a),\mathcal{T}}, a)$ by restricting h to the subtree rooted at $\mathbf{d}$, and thus, $(\mathbf{d}, t_{\mathbf{d}}) \in \mathsf{Hom}$. Using the induction hypothesis, we conclude that $(\mathbf{d}, t_{\mathbf{d}}) \in \mathsf{Hom}_{i+1}$. Hence, $(\mathbf{c}, t) \in \mathsf{Hom}_{i+2}$, which finishes the proof of (*).

Now we can show that $\mathsf{Hom} \subseteq \mathsf{Hom}_i$ for all i, again by induction on i. The case $i = 0$ has been shown above, and the induction step follows from (*).

($\supseteq$): Let $(\mathbf{c}, t) \in \bigcap_{i=0}^{\infty} \mathsf{Hom}_i$. We construct a homomorphism $h : (\mathcal{P}_{\mathbf{c}}, \mathbf{c}) \rightarrow_\Sigma (\mathcal{U}_{t(a),\mathcal{T}}, a)$ level by level, i.e. we inductively define a sequence $h_j : (\mathcal{P}_{\mathbf{c}}|_j, \mathbf{c}) \rightarrow_\Sigma (\mathcal{U}_{t(a),\mathcal{T}}, a)$, where $\mathcal{P}_{\mathbf{c}}|_j$ denotes the restriction of $\mathcal{P}_{\mathbf{c}}$ to the first j levels. While constructing the h_j, we will keep the following invariants:

- h_j is a Σ-homomorphism on its domain.
- For every leaf $\mathbf{p}$ of $\mathcal{P}_{\mathbf{c}}|_j$ we have $(\mathsf{tail}(\mathbf{p}), \mathsf{tp}(h_j(\mathbf{p})) \in \bigcap_{i=0}^{\infty} \mathsf{Hom}_i$.

We have $\mathsf{dom}(h_0) = \{\mathbf{c}\}$ and set $h_0(\mathbf{c}) = a$. Note that h_0 is a homomorphism, since $(\mathbf{c}, t) \in \mathsf{Hom}_0$ and the second invariant is also true. Now assume h_j has been defined. To define h_{j+1}, consider any leaf $\mathbf{prd} \in \Delta^{\mathcal{P}_{\mathbf{c}}|_{j+1}}$. From the second invariant we know that $(\mathbf{p}, \mathsf{tp}(h_j(\mathbf{p}))) \in \mathsf{Hom}_i$ for all i. So for every i we have a type t_i such that $(\mathbf{d}, t_i) \in \mathsf{Hom}_i$. Since there are only finitely many types, there must be a type t' that appears infinitely many times among the t_i. Clearly $t \rightsquigarrow_r^{\mathcal{T},\mathcal{A}} t'$, so we can choose this type as the image of $\mathbf{d}$, i.e. we set $h_{j+1}(\mathbf{prd}) = h_j(\mathbf{p})rt'$. Since t' appeared infinitely many times in the sequence t_i, and since the sequence Hom_i is descending, we have $(\mathbf{d}, t') \in \bigcap_{i=0}^{\infty} \mathsf{Hom}_i$, i.e. the second invariant holds for h_{j+1}. The first invariant follows from the fact that $(\mathbf{d}, t') \in \mathsf{Hom}_0$ and because h_j is a Σ-homomorphism on its domain.

Finally, we set $h = \bigcup_{i=0}^{\infty} h_j$. Since $\mathcal{P}_{\mathbf{c}}$ is connected, we have $\mathsf{dom}(h) = \Delta^{\mathcal{P}_{\mathbf{c}}}$ and the first invariant assures that h is a Σ-homomorphism $(\mathcal{P}_{\mathbf{c}}, \mathbf{c}) \rightarrow_\Sigma (\mathcal{U}_{t(a),\mathcal{T}}, a)$, so $(\mathbf{c}, t) \in \mathsf{Hom}$. This finishes the proof of the claim.

Now we argue the time complexity of computing Hom. Since $\mathsf{Hom}_{i+1} \supseteq \mathsf{Hom}_i$ for all i and since Hom_0 contains at most $|\Delta^k \times \mathsf{TP}|$ many elements, which is single exponential measured in the size of $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$, the sequence Hom_i stabilizes after exponentially many steps. Computing Hom_0 and computing Hom_{i+1} from Hom_i can also be done in exponential time. Hence, Hom can be decided in ExpTime. □

Now we are ready to settle the complexity of QBE(Horn-$\mathcal{ALC}$, CQ).

Theorem 4.11. *For $\mathcal{L}$ = Horn-$\mathcal{ALC}$ and Q = CQ, the problems* QBE*$(\mathcal{L}, Q)$,* QBE*$_f(\mathcal{L}, Q)$,* QDEF*$(\mathcal{L}, Q)$ and* QDEF*$_f(\mathcal{L}, Q)$ are all* coNExpTime*-complete.*

Proof. Given $\mathcal{T}, \mathcal{A}, S^+$ and possibly S^- and/or Σ (depending on the variation of the problem), we need to check the two conditions listed in Theorem 4.4. By Lemma 4.6, we can focus on deciding the second condition. By Lemma 4.8 and Lemma 4.9, this can be done as follows: Guess a $\mathbf{b} \in S^-$ (or in the case of QDEF, a $\mathbf{b} \in \mathsf{ind}(\mathcal{A})^n \setminus S^+$), a Σ-homomorphism $h : (\mathcal{P}|_N, \mathbf{a}^*) \rightarrow_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ and a labelling $L(\pi) \subseteq \Delta^k$ for every $\pi \in \mathsf{range}(h) \cup \mathsf{ind}(\mathcal{A})$. This can be done by a nondeterministic Turing machine in exponential time. Verify that Conditions 1 to 3 are fulfilled. For Conditions 1, 2 and

3a), it is clear that this can be done in exponential time. Condition 3b) can be decided in exponential time by Lemma 4.10. If all conditions are fulfilled, then the instance no-instance, otherwise it is a yes-instance. Thus, we have a coNExpTime algorithm.

A matching coNExpTime lower bound for all four variations is inherited from the database setting [CD15]. □

For UCQs, a careful analysis of Lemma 4.9 yields an ExpTime upper bound; the matching lower bound is obtained by a reduction from subsumption in Horn-$\mathcal{ALC}$ [KRH13].

Theorem 4.12. *For $\mathcal{L}$ = Horn-$\mathcal{ALC}$ and $\mathcal{Q}$ = UCQ, the problems* QBE$(\mathcal{L},\mathcal{Q})$*,* QBE$_f(\mathcal{L},\mathcal{Q})$*,* QDEF$(\mathcal{L},\mathcal{Q})$ *and* QDEF$_f(\mathcal{L},\mathcal{Q})$ *are all ExpTime-complete.*

The ExpTime-completeness follows immediately from the following two lemmas.

Lemma 4.13. *Both* QBE$_f$*(Horn-$\mathcal{ALC}$, UCQ) and* QDEF$_f$*(Horn-$\mathcal{ALC}$, UCQ) are ExpTime-hard.*

Proof. We reduce from (the complement of) the subsumption problem in Horn-$\mathcal{ALC}$, which is the problem of deciding, given a Horn-$\mathcal{ALC}$-TBox $\mathcal{T}$ and concept names A, B, whether $\mathcal{T} \not\models A \sqsubseteq B$. The problem is ExpTime-complete [KRH13].

Given a TBox $\mathcal{T}$ and concept names A, B, define an input $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ for QBE$_f$, by taking $\mathcal{A} = \{A(a), B(b)\}$, $S^+ = \{b\}$ and $S^- = \{a\}$. We now argue correctness of the reduction. It is easy to see that $\mathcal{T} \models A \sqsubseteq B$ if and only if there is a homomorphism $h : \mathcal{U}_{\{B(b)\},\mathcal{T}} \to \mathcal{U}_{\{A(a)\},\mathcal{T}}$ with $h(b) = a$. By Theorem 4.5, this is that case if and only if $(\mathcal{T}, \mathcal{A}, S^+, S^-) \notin$ QBE$_f$. (Note that the constructed instance is always safe and that both S^+ and S^- contain only one individual each.) Thus, we have $\mathcal{T} \not\models A \sqsubseteq B$ if and only if $(\mathcal{T}, \mathcal{A}, S^+, S^-) \in$ QBE$_f$. The ExpTime-hardness of QDEF$_f$ is obtained by the same reduction, ignoring S^-. □

Lemma 4.14. *Both* QBE*(Horn-$\mathcal{ALC}$, UCQ) and* QDEF*(Horn-$\mathcal{ALC}$, UCQ) are in ExpTime.*

Proof. We only describe an exponential time algorithm for QBE(Horn-$\mathcal{ALC}$, UCQ); the arguments for QDEF are essentially the same. By Theorem 4.4, we need to check Σ-safety of $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ for every $\mathbf{a} \in S^+$ and whether $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \not\to (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ for all $\mathbf{a} \in S^+$ and $b \in S^-$. The former can be checked in ExpTime by Lemma 4.6. For the latter, we loop over all (polynomially many) pairs $(\mathbf{a}, \mathbf{b}) \in S^+ \times S^-$ and use Lemma 4.9 for every such pair individually, that is, in the definition of $\mathcal{P}$ we use $k = 1$. In this case, there are only single exponentially many candidates for the homomorphism h and the labeling T, so we can loop over all possible h and T and check Conditions 1 to 3. Clearly, Conditions 1, 2 and 3a can be checked in ExpTime, whereas 3b can be checked in ExpTime by Lemma 4.10. □

4.3.2 $\mathcal{ELI}$ (Undecidability)

In this section, we prove that by allowing inverse roles in the TBox, QBE becomes undecidable. In particular, we show the following:

Theorem 4.15. *For $\mathcal{L} = \mathcal{ELI}$ and $\mathcal{Q} = \text{CQ}$, the problems* QBE$(\mathcal{L}, \mathcal{Q})$, QBE$_f(\mathcal{L}, \mathcal{Q})$, QDEF$(\mathcal{L}, \mathcal{Q})$ *and* QDEF$_f(\mathcal{L}, \mathcal{Q})$ *are all undecidable.*

As noted earlier, this theorem refutes the claim in [GJS18b] that these problems are decidable. The proof is very similar to the undecidability result for the problem concept-by-example for $\mathcal{ELI}$ knowledge bases [Fun+19], where the questions is whether two individuals can be distinguished by an $\mathcal{ELI}$ concept.

The proof is by reduction of the rectangle tiling problem. An instance of the *rectangle tiling problem* is a tuple (T, H, V, t_I, t_F) where T is a finite set of *tile types*, $H, V \subseteq T \times T$ are the *horizontal and vertical compatibility relations*, and $t_I, t_F \in T$ are the *initial and final tile*. A *solution* consists of a tiling τ of some $n \times m$-grid, $n, m \geq 1$, that is, a function $\tau : \{1, \ldots, n\} \times \{1, \ldots, m\} \to T$ such that the following conditions are satisfied:

1. $\tau(1,1) = t_I$ and $\tau(n,m) = t_F$;
2. $(\tau(i,j), \tau(i+1,j)) \in H$ for $1 \leq i < n$ and $1 \leq j \leq m$;
3. $(\tau(i,j), \tau(i,j+1)) \in V$ for $1 \leq i \leq n$ and $1 \leq j < m$.

We assume that T is partitioned into $T_0 \uplus T_1 \uplus T_2$ and that the following conditions are satisfied:

C1 if $(t,t') \in H$ and $t \in T_i$, $i \in \{0,1,2\}$, then $t' \in T_{i+1 \bmod 3}$;
C2 if $(t,t') \in V$ and $t \in T_i$, $i \in \{0,1,2\}$, then $t' \in T_i$;
C3 $t_I \in T_0$ and $t_F \in T_2$.
C4 t_F can only be used in the upper right corner, that is neither H nor V contains a pair of the form (t_F, t);
C5 there is a unique tile t'_F that must be placed to the left of t_F and cannot be used anywhere else, that is, $(t'_F, t_F) \in H$, $(t'_F, t) \in H$ implies $t = t_F$, and $(t, t_F) \in H$ implies $t = t'_F$.

We briefly argue why the rectangle tiling problem is undecidable. Given a Turing machine, one constructs a set of tiles which can be used to describe the computation tableau of the Turing machine. Every tile describes the content of a tape cell at a certain time, a row of a solution corresponds to a configuration of the Turing machine. The tile t_F indicates the last (right-most) cell of a halting configuration. Thus, the constructed instance of the rectangle tiling problem has a solution if and only if the Turing machine halts. It is easy to show that Conditions C1 to C5 can be assumed as well. To avoid dealing with special cases, we also assume that if there is a tiling of some $n \times m$-grid, then there is a tiling of an $n \times m$-grid with $m > 2$. Note that, due to the assumed conditions, all tiles on the left-most column must be from T_0 and all tiles on the right-most column must be from T_2. Moreover, n must be divisible by 3.

We focus on undecidability of QBE$_f(\mathcal{ELI}, \text{CQ})$, undecidability of the other three problems will follow easily. Let $\mathcal{P} = (T, H, V, t_I, t_F)$ be an instance of the rectangle tiling problem. We construct an instance $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ such that $\mathcal{P}$ has a solution if and only if $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ is a yes-instance of QBE, if and only if, by Theorem 4.4, $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a}) \to (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ for some $\mathbf{b} \in S^-$. (We guarantee safety of $\prod_{\mathbf{a} \in S^+}(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{a})$ in our construction.) For the rest of this section, let $\mathcal{U}$ denote $\mathcal{U}_{\mathcal{A},\mathcal{T}}$.

Set $\mathcal{A} = \{P_1(a_1), P_2(a_2), N_1(b), N(a_1), N(a_2), N(b)\}$, $S^+ = \{a_1, a_2\}$, and $S^- = \{b\}$. The concept names P_1, P_2, N_1 trigger the construction of different trees below a_1, a_2, b in $\mathcal{U}$, via the TBox $\mathcal{T}$ that is at the heart of the construction. The concept name N asserted at all three individuals has technical reasons that can be ignored for now. Roughly speaking, certain paths in the product $(\mathcal{U}, a_1) \otimes (\mathcal{U}, a_2)$ starting at (a_1, a_2) should correspond to solutions of the tiling. We first explain the symbols used in $\mathcal{T}$:

- a single reflexive and symmetric role name S, represented via the role composition $r^-; r$; that is, we use $\exists S.C$ as an abbreviation for $\exists r^-.\exists r.C$;
- for each tile type $t \in T$, three concept names B^0_t, B^1_t, B^2_t; additionally, concept names B^0_d, B^1_d, B^2_d where $d \notin T$ is a dummy tile;
- a concept name E that marks the last node of the first row in a row by row traversal of the grid, from bottom to top;
- a concept name N that marks intermediate nodes inserted between any two rows in the traversal;
- to avoid interaction between the different trees and control the construction of the universal model, auxiliary concept names I, F and X, which only appear in the tree below a_1, and auxiliary concept name G, which only appears in the tree below a_2.

We start with the tree rooted at a_1, writing $i \oplus k$ as an abbreviation for $i + k \bmod 3$:

1. $P_1 \sqsubseteq \displaystyle\bigsqcap_{(t_I,t)\in V} \exists S.(N \sqcap \exists S.(B^0_{t_I} \sqcap B^1_t \sqcap I))$

2. for all $t_1, t_2 \in T_j \setminus \{t_F\}$, $j \in \{0, 2\}$:
$$B^0_{t_1} \sqcap B^1_{t_2} \sqcap I \sqsubseteq \bigsqcap_{(t_1,t_3)\in H,(t_3,t_4)\in V} \exists S.(B^0_{t_3} \sqcap B^1_{t_4} \sqcap I)$$

3. for all $t_1, t_2 \in T_1$:
$$\begin{aligned} B^0_{t_1} \sqcap B^1_{t_2} \sqcap I \ \sqsubseteq \ & \bigsqcap_{\substack{(t_1,t_3)\in H,\\(t_3,t_4)\in V}} \exists S.(B^0_{t_3} \sqcap B^1_{t_4} \sqcap I) \sqcap \\ & \bigsqcap_{\substack{(t_1,t_3)\in H,\\(t_3,t_4)\in V}} \exists S.(B^0_{t_3} \sqcap B^1_{t_4} \sqcap E) \end{aligned}$$

4. for all $t_1, t_2 \in T_2 \setminus \{t_F\}$:
$$B^0_{t_1} \sqcap B^1_{t_2} \sqcap E \sqsubseteq \bigsqcap_{t_3\in T_0,(t_3,t_4)\in V} \exists S.(N \sqcap \exists S.(B^1_{t_3} \sqcap B^2_{t_4} \sqcap X))$$

5. for all $i \in \{0,1,2\}$ and $t_1, t_2 \in T_j$, $j \in \{0,1\}$:

$$B^i_{t_1} \sqcap B^{i\oplus 1}_{t_2} \sqcap X \sqsubseteq \bigsqcap_{(t_1,t_3)\in H,(t_3,t_4)\in V} \exists S.(B^i_{t_3} \sqcap B^{i\oplus 1}_{t_4} \sqcap X)$$

6. for all $i \in \{0,1,2\}$ and $t_1, t_2 \in T_2 \setminus \{t_F\}$:

$$\begin{aligned} B^i_{t_1} \sqcap B^{i\oplus 1}_{t_2} \sqcap X \quad \sqsubseteq & \bigsqcap_{\substack{(t_1,t_3)\in H,\\(t_3,t_4)\in V}} \exists S.(B^i_{t_3} \sqcap B^{i\oplus 1}_{t_4} \sqcap X) \sqcap \\ & \bigsqcap_{\substack{t_3\in T_0,\\(t_3,t_4)\in V}} \exists S.(N \sqcap \exists S.(B^{i\oplus 1}_{t_3} \sqcap B^{i\oplus 2}_{t_4} \sqcap X)) \end{aligned}$$

7. for all $i \in \{0,1,2\}$ and $t \in T_2$:

$$B^i_t \sqcap B^{i\oplus 1}_{t_F} \sqcap X \quad \sqsubseteq \quad \bigsqcap_{t'\in T_0} \exists S.(N \sqcap \exists S.(B^{i\oplus 1}_{t'} \sqcap F))$$

8. for all $i \in \{0,1,2\}$ and $t \in T \setminus \{t'_F, t_F\}$:

$$B^i_t \sqcap F \quad \sqsubseteq \quad \bigsqcap_{(t,t')\in H} \exists S.(B^i_{t'} \sqcap F)$$

9. for all $i \in \{0,1,2\}$:

$$B^i_{t'_F} \sqcap F \quad \sqsubseteq \quad B^i_d \sqcap \exists S.B^i_{t_F}$$

A *tiling word* is a word over the alphabet $T \cup \{N\}$. Let τ be the tiling of some $n \times m$-grid. The *row by row unfolding of* τ is the tiling word

$$\tau(1,1)\cdots\tau(n,1)N\cdots N\tau(1,m)\cdots\tau(n,m).$$

Note that we use the symbol N to separate the rows. The concept inclusions above generate a tree in which for every tiling τ of some $n \times m$-grid, we we find a path p that describes the row by row unfolding of τ. This is even true if the third condition of tilings is not satisfied. Here and in what follows, a *path* is a sequence of domain elements $p = d_0 \cdots d_n$ such that

$$(d_i, d_{i+1}) \in S^{\mathcal{U}} := (r^-)^{\mathcal{U}} \circ r^{\mathcal{U}}$$

for all $i < n$. Each element d on p is labelled with N or with two concept names B^i_t and $B^{i\oplus 1}_{t'}$ with $(t,t') \in V$ to indicate that the grid position represented by d carries tile t' and that the grid position directly below the position represented by d carries tile t. The first part of p (between the first two occurrences of N) uses concept names B^0_t and $B^1_{t'}$ and gives the tiling of rows 0 and 1. It's last element satisfies the concept name E. The next part of p uses concept names B^1_t and $B^2_{t'}$ and gives the tiling of (row 1 once again and of) row 2. And so on, modulo 3. The horizontal tiling condition is satisfied on the entire path.

After the last part of the path, which gives the tiling of the topmost row and repeats the tiling of the row below it, there is another segment that is labelled with F and in which each node is labelled only with a single concept name B^i_t, repeating the labelling of the topmost row. A notable difference is that the position before the last one is not only labelled with a concept name $B^i_{t'_F}$, but also with B^i_d. The last element on that segment is a leaf, that is, it has no successors.

We next define the tree rooted at a_2:

10. $P_2 \sqsubseteq E \sqcap \bigsqcap_{t \in T_0} \exists S.(N \sqcap \exists S.(B^1_t \sqcap G))$

11. for all $i \in \{0,1,2\}$ and $t \in T_j \setminus \{t'_F\}, j \in \{0,1\}$:

$$B^i_t \sqcap G \sqsubseteq \bigsqcap_{t' \in T_{j+1}} \exists S.(B^i_{t'} \sqcap G)$$

12. for all $i \in \{0,1,2\}$ and $t \in T_2 \setminus \{t_F\}$:

$$\begin{aligned} B^i_t \sqcap G \quad \sqsubseteq \quad & \bigsqcap_{t' \in T_0} \exists S.(B^i_{t'} \sqcap G) \sqcap \\ & \bigsqcap_{t' \in T_0} \exists S.(N \sqcap \exists S.(B^{i \oplus 1}_{t'} \sqcap G)) \end{aligned}$$

13. for all $i \in \{0,1,2\}$:

$$B^i_{t'_F} \sqcap G \sqsubseteq \exists S.(B^i_{t_F} \sqcap \exists S.(N \sqcap \bigsqcap_{t \in (T \cup \{d\}) \setminus \{t_F, t'_F\}} B^i_t))$$

The generated tree contains every path p on which every element is labelled with N or with a single concept names B^i_t, subject to the following conditions. The path starts with an N. The part between the first two occurrences with N is labelled with concept names B^1_t, the part between the second two occurrences with concept names B^2_t, and so on. Moreover, Condition C1 must be respected. Nodes labelled with a concept $B^i_{t'_F}$ are special. They have a successor d that we call a *pre-cycle node* and that satisfies $B^i_{t_F}$ (and no other successor). The pre-cycle node d has as its (only) successor a leaf node d' that we call a *cycle node* that satisfies B^i_t for all $t \in T$ except t_F and t'_F, and also N and B^i_d. This part of the TBox also labels a_2 with E. Informally, the E-labeling in the tree below a_2 is offset by -1 row compared to the E-labeling in the tree below a_1, and this plays a central role in the reduction.

For the tree rooted at b, define a set C of concept names as

$$C = \{N\} \cup \{B^j_t \mid j \in \{0,1,2\} \text{ and } t \in T \cup \{d\}\}.$$

and include the following concept inclusion in $\mathcal{T}$:

14.
$$N_1 \sqsubseteq \exists S.(\bigsqcap_{A \in C} A \sqcap \exists S.\exists S.(E \sqcap \bigsqcap_{A \in C} A))$$

The universal model generated below b, shown in Figure 4.1, has the property the every path $d_0 \dots d_n$ where every individual on the path satisfies a concept name from C can be homomorphically embedded by a homomorphism that maps all d_i to the S-successor of b, remember that S is reflexive and symmetric. A path $d_0 \dots d_n$ containing the concept name E however, say d_j is labelled with E, can be embedded if and only if the path contains a *hole* before the E, that is, an individual d_i on the path, $i < j$, such that d_i does not satisfy any concept name.

The correctness of the reduction is based on the idea that a solution for the tiling problems exists if and only if we can find a path $d_0 \dots d_n$ in $\mathcal{U} \times \mathcal{U}$ such that d_0 is a successor of (a_1, a_2), d_n is labelled with E and the path does not contain a hole, that is, every individual on the path is labelled with some concept name. The following lemma says that we can in fact limit our attention to path-shaped CQ witnesses.

Lemma 4.16. *If the constructed instance $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ is a yes-instance of* QBE$(\mathcal{ELI}, \text{CQ})$, *then there exists a witnessing CQ $q'(x)$ with the following properties:*

- *$q'(x)$ is takes the form of an S-path from x to a variable labelled with E.*
- *No variable on the S-path $q'(x)$ is a hole.*

Proof. Let $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ be a yes-instance of QBE$(\mathcal{ELI}, \text{CQ})$, so there exists a CQ $q(x)$ with $\mathcal{T}, \mathcal{A} \models q(a_i)$ for $i \in \{1, 2\}$ and $\mathcal{T}, \mathcal{A} \not\models q(b)$. We identify an S-path-shaped subset $q'(x)$ of the body of $q(x)$ that fulfils the conditions from the lemma.

Towards a contradiction, assume that there is no such subset $q'(x)$ of $q(x)$. We show that this would give a homomorphism h from $q(x)$ to $\mathcal{U}$ with $h(x) = b$, contradicting $\mathcal{T}, \mathcal{A} \not\models q(b)$. We can assume q to be connected, and by construction of the universal model below a_1 and below a_2, q can only use concept names from C and the concept name E, since these are the only concept names that appear both below a_1 and below a_2. Further, we can assume w.l.o.g. that all S-neighbours of x are labelled with N, since all S-neighbours of (a_1, a_2) in $\mathcal{U}$ are labelled with N. This means that no S-neighbour of x is a hole, a fact that we need soon. In fact, this is the only reason why we defined $\mathcal{A}$ to contain $N(a_1), N(a_2), N(b)$. We partition all variables in $q(x)$ that are reachable by an S-path from x (but excluding x itself) into three sets B_1, B_2, B_3, as follows:

- B_1 is the set of all non-holes y for which there is an S-path from x to y that does not have a hole;
- B_2 is the set of all holes that are S-neighbours of elements in B_1;
- B_3 is the set of all y such that all S-paths from x to y have a hole different from y.

Note that B_1, B_2, B_3 indeed form a partition of all elements reachable via an S-path from x. Moreover, B_2 separates B_1 and B_3 in the sense that every S-path from an element in B_1 to an element in B_3 must pass an element of B_2; in the same way, B_1 separates x and B_2, because no S-neighbour of x is a hole.

Define h for elements from B_i and x as follows:

- $h(x) = b$;
- for all $i \in \{1,2,3\}$ and all $y \in B_i$, set $h(y) = b_i$, where b_i is the anonymous i-times S-successor of b, see Figure 4.1.

It should be clear that $A(y) \in q$ implies $h(y) \in A^{\mathcal{U}}$ for every concept name $A \in \mathsf{C} \cup \{E\}$ due to the construction of the universal model below b. In particular this is true for the concept name E, since our assumption is that every path from the answer variable to a variable labelled with E contains a hole, so all variables labelled with E have to be in B_3. Moreover, we have that

(∗) $S(y_1, y_2) \in q$ implies $(h(y_1), h(y_2)) \in S^{\mathcal{U}}$.

It remains to define h on the intermediate variables of S-paths, that is, r^--successors of variables reachable via an S-path from x. Let y be such a variable. Consider the set Y of all r-successors of y. Note that h is already defined for all variables from Y, and we have $S(y_1, y_2) \in q$, for all $y_1, y_2 \in Y$. By (∗), we have $(h(y_1), h(y_2)) \in S^{\mathcal{U}}$, for all $y_1, y_2 \in Y$. Thus Y is completely contained in one of the sets $\{b, b_1\}$, $\{b_1, b_2\}$ or $\{b_2, b_3\}$. Depending on which case applies, we complete the definition of h by setting $h(y)$ to the intermediate nodes between b and b_1, b_1 and b_2, and b_2 and b_3, respectively. This yields a homomorphism h witnessing $\mathcal{T}, \mathcal{A} \models q(b)$, a contradiction. Thus, there must be an S-path in q as claimed. □

We are now ready to prove correctness of the reduction. We give a brief explanation of this proof up front. If there is a witness CQ, then there is also a S-path-shaped witness CQ fulfilling the conditions of Lemma 4.16. The construction assures that the labelling of such a path in the product $\mathcal{U} \times \mathcal{U}$, starting at (a_1, a_2), must describe a row by row unfolding of a solution of the rectangle tiling problem. In particular, to reach an individual labelled with E in the product, via an S-path without holes, we are forced to walk downwards from both individuals a_1 and a_2 for a while, then cycle below a_2 while walking down further below a_1, and then walking upwards in both trees again. The row by row unfolding of the solution is already contained in the part of the path where we walk downwards in both components. The cycling in the second component is done to obtain an offset of 1 row between the two components, so that the part of the path where we walk upwards (and still have no holes) guarantees that the rows are of the same length and that the vertical matching condition for the rows is fulfilled. This is the reason why the elements in the tree below a_1 are always labelled with two vertically matching tiles.

Lemma 4.17. *The tiling problem $\mathcal{P}$ has a solution if and only if $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ is a yes-instance of* QBE$(\mathcal{ELI}, \text{CQ})$.

Proof. Consider two elements $d, d' \in \Delta^{\mathcal{U}}$ that are both part of the subtree in $\mathcal{U}$ rooted at a_i, $i \in \{1,2\}$, and such that $(d, d') \in S^{\mathcal{U}}$. We call d' a *successor* of d if d' is further away from a_i than d, and the *predecessor* of d if d is further away from a_i than d'.

"if". Let $(\mathcal{T}, \mathcal{A}, S^+, S^-)$ be a yes-instance. Using Lemma 4.16, let $q(x)$ be a CQ of the kind assured in the lemma. The matches of q into $\mathcal{U}$ that witness $\mathcal{T}, \mathcal{A} \models q(a_i)$ for

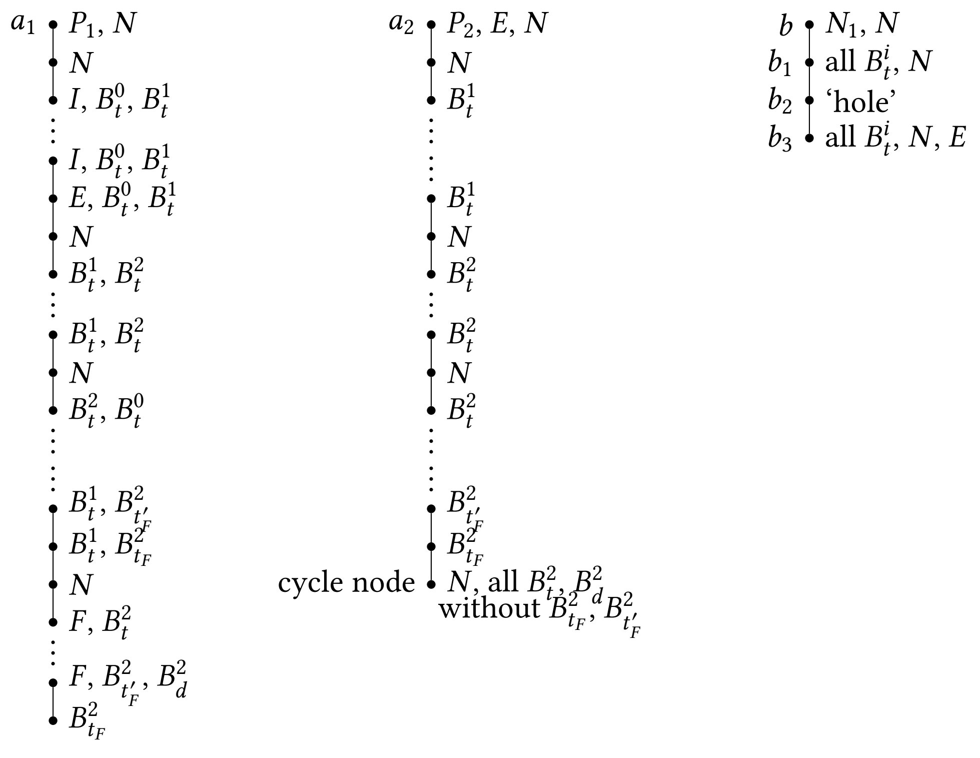

Figure 4.1: The image shows two paths in $\mathcal{U}$. Every drawn edge is an S-edge.

$i \in \{1, 2\}$ yield the existence of a path $p = (d_0, e_0) \cdots (d_n, e_n)$ in $\mathcal{U} \times \mathcal{U}$ that starts at an S-successor of (a_1, a_2) and such that (d_n, e_n) satisfies E in $\mathcal{U} \times \mathcal{U}$ and p has no holes, that is, for all $i < n$, (d_i, e_i) satisfies in $\mathcal{U} \times \mathcal{U}$ at least one concept name from C. We might clearly assume that p is a simple path, that is, $i \neq j$ implies $(d_i, e_i) \neq (d_j, e_j)$.

By construction of the trees below a_1 and a_2, every node (d_i, e_i) on p satisfies a unique concept name from C. We will later show how to read off from this unique labelling of p a tiling word that is a row by row unfoldung of a tiling of some finite grid.

It is important to carefully analyse how p lies within $\mathcal{U} \times \mathcal{U}$. We say that (d_{i+1}, e_{i+1}) is a $\downarrow\downarrow$-*successor* of (d_i, e_i) if d_{i+1} is a successor of d_i and e_{i+1} is a successor of e_i, and likewise for $\downarrow\uparrow$-*successors*, $\downarrow\circlearrowleft$-*successors*, and so on, with $\uparrow$ indicating the transition to a predecessor in the respective component and $\circlearrowleft$ indicating identity of the component (recall that S is reflexive).

We make one observation about the very beginning of the path p: We can assume w.l.o.g. that (d_0, e_0) and (d_1, e_1) are both $\downarrow\downarrow$-successors, which can be argued as follows: By the construction, (d_0, e_0) must be labelled with N. Consider the largest number j such that $(d_0, e_0) \dots (d_j, e_j)$ are all labelled with N. Every d_i and e_i, $0 \leq i \leq j$ must be either a_1, a_2 or an S-successor of a_1 or of a_2. Also, $d_j \neq a_1$ and $e_j \neq a_2$, since otherwise, (d_{j+1}, e_{j+1}) also had to be labelled with N. If now $j > 0$, instead of p, we could consider the path $p' = (d'_0, e'_0)(d'_1, e'_1) \dots$ with $(d'_i, e'_i) = (d_{i+j}, e_{i+j})$ for all $i \geq 2$, which is a path without holes starting from a $\downarrow\downarrow$-successor of (a_1, a_2) to an element labelled with E. Figuratively speaking, p' cuts short the initial part of p labelled with unnecessary Ns. So from now, we assume that p is such a path.

Next, we observe that several kinds of successors cannot occur on p:

(i) $\uparrow\downarrow$-successors (d_{i+1}, e_{i+1}) with e_{i+1} not a cycle node.

Towards a proof by contradiction, assume that (d_{i+1}, e_{i+1}) is an $\uparrow\downarrow$-successor with e_{i+1} not a cycle node. We know that (d_i, e_i) satisfies a concept name from C. Since e_{i+1} is not a cycle node, this concept name is not of the form B^j_d. First assume that it has the form B^j_t, $t \in T$. Let $t \in T_w$. Both d_i and e_i also satisfy B^j_t. By construction of the trees below a_1 and a_2 and since (d_{i+1}, e_{i+1}) is an $\uparrow\downarrow$-successor of (d_i, e_i),

- d_{i+1} satisfies N or some $B^\ell_{t'}$ with $t' \in T_{w\ominus 1}$, but no other tile from C except possibly B^ℓ_d;
- e_{i+1} satisfies N or some $B^{\ell'}_{t'}$ with $t' \in T_{w\oplus 1}$ and no other concept name from C;
- d_{i+1} and e_{i+1} do not both satisfy N (as there are at least three non-N-nodes between any two consecutive N-nodes in $\mathcal{U}$).

As a consequence, (d_{i+1}, e_{i+1}) is a hole. Contradiction.

Now assume that (d_i, e_i) satisfies N. Then so do d_i and e_i. By construction of the trees below a_1 and a_2 and since e_{i+1} is not a cycle node, d_{i+1} satisfies some $B^\ell_{t'}$ with $t' \in T_2$ and no other concept name from C, and e_{i+1} satisfies some $B^{\ell'}_{t'}$ with $t' \in T_0$ and no other concept name from C. As a consequence, (d_{i+1}, e_{i+1}) is a hole. Contradiction.

(ii) $\circlearrowleft *$-successors.

First for the $\circlearrowleft\downarrow$ case. Towards a proof by contradiction, assume that (d_{i+1}, e_{i+1}) is a $\circlearrowleft\downarrow$-successor of (d_i, e_i). First assume that e_{i+1} is not a cycle node. Then e_i and e_{i+1} satisfy unique but different concept names from C that are not of the form B^ℓ_d. The first such concept name is also satisfied by (d_i, e_i), thus by d_i, and the second concept name is also satisfied by (d_{i+1}, e_{i+1}), thus by $d_{i+1} = d_i$. But by construction of the tree below a_1, d_i does not satisfy two such different concept names. Contradiction.

Now assume that e_{i+1} is a cycle node. By construction of the subtree below a_2, e_i satisfies a concept name of the form $B^j_{t_F}$, and thus so does d_i. Moreover, (d_{i+1}, e_{i+1}) satisfies a concept name from C also satisfied by e_{i+1} and since e_{i+1} is a cycle node, this concept name cannot be of the form $B^j_{t_F}$ or $B^j_{t'_F}$. Thus $d_{i+1} = d_i$ satisfies both concept names, which is not possible by the construction of the tree below a_1.

The case $\circlearrowleft\uparrow$ is similar. Furthermore, there are no $\circlearrowleft\circlearrowleft$-successors since p is simple.

(iii) $\uparrow\circlearrowleft$-successors (d_{i+1}, e_{i+1}) with (d_i, e_i) not an $\uparrow\downarrow$-successor.

Towards a proof by contradiction, assume that (d_{i+1}, e_{i+1}) is a $\uparrow\circlearrowleft$-successor of (d_i, e_i) with (d_i, e_i) not an $\uparrow\downarrow$-successor, and that it is the first such node on p. If e_i is not a cycle node, then we can argue as in (ii) above. Thus assume that e_i is a cycle node. Consider the successor type of (d_i, e_i). Since e_i is a cycle node, it is a leaf in $\mathcal{U}$. Together with Point (ii) above and since (d_i, e_i) is the first $\uparrow\circlearrowleft$-successor, (d_i, e_i) can thus only be a $\downarrow\circlearrowleft$-successor or a $\downarrow\downarrow$-successor. The former implies $(d_{i-1}, e_{i-1}) = (d_{i+1}, e_{i+1})$ in contradiction to p being simple. In the latter case, $d_{i-1} = d_{i+1}$ and e_{i-1} is the predecessor of e_i in $\mathcal{U}$. By construction of the tree below a_1, the latter implies that e_{i-1} is labelled with some concept name $B^j_{t_F}$. Since (d_{i-1}, e_{i-1}) is not a hole in p, d_{i-1} is also labelled with $B^j_{t_F}$ and thus so is $d_{i+1} = d_{i-1}$. However, by construction of the subtree below a_2, the cycle node e_{i+1} is not labelled with $B^j_{t_F}$ and thus (d_{i+1}, e_{i+1}) is a hole in p. Contradiction.

(iv) $\downarrow\uparrow$-successors (d_{i+1}, e_{i+1}) with e_i not a cycle node.

Similar to the the proof of (i).

(v) $\downarrow\circlearrowleft$-successors (d_{i+1}, e_{i+1}) with e_i not a cycle node.

Similar to the proof of (ii).

The remaining kinds of successors are $\downarrow\downarrow$, $\uparrow\uparrow$, $\uparrow\downarrow$, $\uparrow\circlearrowleft$, $\downarrow\uparrow$, $\downarrow\circlearrowleft$, and Points (i) to (v) impose strong restrictions on the latter four types of successors. We aim to show that p must follow the pattern

$$\downarrow\downarrow^+ \downarrow\circlearrowleft^+ \downarrow\uparrow\ \uparrow\uparrow^+$$

with the $\downarrow\circlearrowleft^+$-subpath having a cycle node in the second component and the $\downarrow\uparrow\ \uparrow\uparrow$-subpath escaping from that cycle node and its pre-cycle node back to a regular node.[1]

We already argued that p must start with a $\downarrow\downarrow^+$-prefix. Assume towards a proof by contradiction that the $\downarrow\downarrow^+$-prefix is followed by an $\uparrow\downarrow$-successor (d_{i_0+1}, e_{i_0+1}). By Point (i), e_{i_0+1} is a cycle node. By construction of the tree below a_2, e_{i_0} must be labelled with a concept name $B^j_{t_F}$. Thus, d_{i_0} is also labelled with $B^j_{t_F}$ and by construction of the subtree below a_1, $d_{i_0+1} = d_{i_0-1}$ is labelled with $B^j_{t'_F}$. In fact, d_{i_0-1} is the first element on the path $d_0, \ldots, d_{i_0-1}$ in $\mathcal{U}$ that is labelled with a concept name of the form $B^\ell_{t'_F}$: if some d_s with $s < i_0 - 1$ was the first element on the path $d_0, \ldots, d_{i_0-1}$ labelled with $B^\ell_{t'_F}$, then e_{s+2} is a cycle node due to the construction of the tree below a_2 and since we travel $\downarrow\downarrow^2$ from (d_s, e_s) to (d_{s+2}, e_{s+2}); this contradicts the fact that e_{i_0} is reachable from e_s by traveling downwards. As d_{i_0-1} is the first element of its kind, it is not labelled with B^ℓ_d and in fact $B^\ell_{t'_F}$ is the only concept name from C satisfied by $d_{i_0-1} = d_{i_0+1}$. However, the cycle node e_{i_0+1} is not labelled with $B^j_{t'_F}$, thus (d_{i_0+1}, e_{i_0+1}) is a hole in p. Contradiction.

Now assume that the $\downarrow\downarrow^+$-prefix is followed by a $\downarrow\uparrow$-successor (d_{i_0+1}, e_{i_0+1}). Then, e_{i_0} is a cycle node by Point (iv) and thus e_{i_0+1} is a pre-cycle node and actually $e_{i_0+1} = e_{i_0-1}$. Thus (d_{i_0-1}, e_{i_0-1}) and (d_{i_0+1}, e_{i_0+1}) both satisfy a concept name $B^j_{t_F}$ and also d_{i_0-1} and d_{i_0+1} both satisfy $B^j_{t_F}$. This, however, is impossible by construction of the subtree below a_1 and in particular due to the partitioning of T into $T_0 \uplus T_1 \uplus T_2$.

[1]This also implies that neither $\uparrow\downarrow$- nor $\uparrow\circlearrowleft$-successors occur at all, but we are not yet in a position to show this directly.

We have thus shown that p starts with a $\downarrow\downarrow^{+}\downarrow\circlearrowleft^{+}$-prefix. By Point (v), the first $\downarrow\circlearrowleft$-successor (d_{i_0+1}, e_{i_0+1}) is such that e_{i_0} is a cycle node. By construction of the subtree below a_2, (d_{i_0-1}, e_{i_0-1}) satisfies a concept name $B^j_{t_F}$ and (d_{i_0-2}, e_{i_0-2}) satisfies $B^j_{t'_F}$. This will be used later.

Since e_{i_0} is a leaf in $\mathcal{U}$, this prefix can only be followed by a successor of type $\downarrow\uparrow$, $\uparrow\uparrow$, and $\uparrow\circlearrowleft$, say (d_{i_1+1}, e_{i_1+1}). However, $\uparrow\circlearrowleft$ is impossible by Point (iii). Moreover, $\uparrow\uparrow$ is impossible too. Assume to the contrary that (d_{i_1+1}, e_{i_1+1}) is an $\uparrow\uparrow$-successor. We know that (d_{i_1-1}, e_{i_1-1}) satisfies a unique concept name from $\mathcal{C}$. Since $e_{i_1-1} = e_{i_1}$ is a leaf node, this concept name is not of the form $B^j_{t_F}$, and it is also satisfied by d_{i_1-1}. But e_{i_1+1} is a pre-cycle node and thus the only concept name it satisfies is of the form $B^j_{t_F}$; the same concept name must be satisfied by $d_{i_1+1} = d_{i_1-1}$. But no element in the subtree below a_1 satisfies two such concept names.

It follows that (d_{i_1+1}, e_{i_1+1}) is a $\downarrow\uparrow$-successor. It thus satisfies a concept name $B^j_{t_F}$. In fact, we have only moved downwards in the first component so far, and thus d_{i_1+1} is the second node on a path in $\mathcal{U}$ that satisfies a concept name $B^j_{t_F}$, the first one being d_{i_0-1}. As a consequence and by construction of the tree below a_1, d_{i_1+1} is a leaf in $\mathcal{U}$ and d_{i_1} satisfies both $B^j_{t'_F}$ and B^j_d. This will be used later.

We next analyze the type of successor that (d_{i_1+2}, e_{i_1+2}) is. Since d_{i_1+1} is a leaf and by Point (iii), the only options are $\uparrow\uparrow$ and $\uparrow\downarrow$. The latter, however is impossible since p is simple. We have shown that p starts with a $\downarrow\downarrow^{+}\downarrow\circlearrowleft^{+}\downarrow\uparrow\ \uparrow\uparrow$-prefix.

We can proceed to travel $\uparrow\uparrow$. We argue that we can never switch to any other kind of successor again. $\uparrow\downarrow$, $\downarrow\uparrow$, and $\downarrow\circlearrowleft$ are ruled out by Points (i), (iv), and (v) and since we can never reach a cycle node in the second component while traveling upwards. $\uparrow\circlearrowleft$ is ruled out by Point (iii). The only remaining candidate is $\downarrow\downarrow$. But we can never switch to $\downarrow\downarrow$ before reaching a_1 in the first component or a_2 in the second component because p is simple. The former cannot happen since a_1 satisfies neither E nor any concept name from C and p has no holes. The latter can (and in fact does) only happen at the final element of p since a_2 satisfies (E but) no concept name from C and thus seeing a_2 before the end means that p has a hole. We have thus shown that p indeed follows the pattern

$$\downarrow\downarrow^{+}\downarrow\circlearrowleft^{+}\downarrow\uparrow\ \uparrow\uparrow^{+}\ .$$

Moreover the last element (d_n, e_n) of p must be such that $e_n = a_2$ because by construction of the tree below a_2 and what we have said about the structure of p, this is the only way for (d_n, e_n) to satisfy E.

To proceed, consider the prefix

$$p' = (d_0, e_0), \dots, (d_{i_0-1}, e_{i_0-1})$$

of p. As already pointed out, each node on p is associated with a unique concept name from $\mathcal{C}$. For the nodes on p', this concept name cannot be of the form B^j_d (recall that d is the dummy tile) since $d_0, \dots, d_{i_0-1}$ constitutes a path in $\mathcal{U}$ that travels purely downwards and sees only one concept name of the form $B^j_{t_F}$ at the very end. In fact, we have already argued that d_{i_0-1} satisfies a concept name $B^j_{t_F}$. No earlier node does so since by construction of the tree below a_2 we would otherwise have reached a cycle node in the second component earlier than at e_{i_0}.

We can thus read off from p' a unique tiling word $t_0 \cdots t_{i_0-1}$. By construction of the trees below a_1 and a_2, $t_0 = N$. Let $t_1 \cdots t_{n_1}$ be the longest prefix of $t_1 \cdots t_{i_0-1}$ that does not contain N. Since (d_1, e_1) is a $\downarrow\downarrow$-successor and again by construction of the trees below a_1 and a_2, this prefix is not empty. Moreover, each node d_i with $1 \leq i \leq n_1$ satisfies a concept name of the form B^0_t and a concept name of the form B^1_t. It is the latter concept that is also satisfied by (d_i, e_i) and thus defines the tiles $t_1 \cdots t_{n_1}$. We obtain another sequence of tiles $t^{(0)}_1 \cdots t^{(0)}_{n_1}$ from the B^0_t labeling. We aim to show that

$$t^{(0)}_1 \cdots t^{(0)}_{n_1} t_0 \cdots t_{i_0-1}$$

is a row by row unfolding of a tiling of some $n_1 \times m$-grid. By construction of the tree below a_1, the following is not hard to verify:

1. $t^{(0)}_1 = t_I$;
2. the horizontal matching condition is satisfied; more formally, whenever tt' is a subword of $t^{(0)}_0 \cdots t^{(0)}_{n_1} t_0 \cdots t_{i_0-1}$ and none of t and t' is N, then $(t, t') \in H$;
3. the first two rows in $t^{(0)}_0 \cdots t^{(0)}_{n_1} N t_0 \cdots t_{n-1}$ are of the same length n_1 and all vertically neighboring tiles on these two rows satisfy V (because the double labeling with B^0_t and $B^1_{t'}$ in the tree below a_1 respects V).

It remains to show that the vertical matching condition is satisfied beyond the first two rows and that all rows rather than only the first two have the intended length n_1.

We associate an *offset* with each (d_i, e_i) on p, defined as the difference $D_2 - D_1$ where D_1 is the distance of d_i from a_1 in $\mathcal{U}$ and D_2 the distance of e_i from a_2 in $\mathcal{U}$. Clearly, the offset of (d_0, e_0) is 0. By construction of the tree below a_1 and choice of n_1, d_{n_1} satisfies E and no other element among $d_0, \ldots, d_{i_0-1}$ does. Moreover, since we first travel only downwards and then only upwards in the first component and d_n satisfies E, we must have $d_n = d_{n_1}$. By construction of the tree below a_2, the only element among $e_0, \ldots, e_n$ satisfying E is $e_n = a_2$. Consequently, the offset of (d_n, e_n) is $n_1 + 1$.

Since the offset of (d_0, e_0) is 0 and until (d_{i_0}, e_{i_0}) we have only seen $\downarrow\downarrow$-successors, the offset of (d_{i_0}, e_{i_0}) must also be 0. Likewise, the offset of (d_n, e_n) being $n_1 + 1$ and the fact that from (d_{i_1+1}, e_{i_1+1}) on we have only seen $\uparrow\uparrow$-successors implies that the offset of (d_{i_1+1}, e_{i_1+1}) must also be $n_1 + 1$. Consequently and since the single $\downarrow\uparrow$-step adds an offset of 2, the $\downarrow\circlearrowleft^+$-subpath of p has length $n_1 - 1$. Clearly, the distance of d_{i_0-1} from a_1 is i_0. To reach (d_{i_1+1}, e_{i_1+1}) from (d_{i_0-1}, e_{i_0-1}), we make one $\downarrow\downarrow$-step, $n_1 - 1$ $\downarrow\circlearrowleft$-steps, and one $\downarrow\uparrow$-step. As a consequence, the distance of d_{i_1+1} from a_1 in $\mathcal{U}$ is $i_0 + n_1 + 1$. Since from (d_{i_1+1}, e_{i_1+1}) we make only $\uparrow\uparrow$-steps and $e_{i_1+1} = e_{i_0-1}$, this implies the following crucial conditions:

(a) if (d_i, e_i) is a node in p with $i < i_0$, then (d_{i+n_1+1}, e_i) is also a node in p;

(b) if (d_i, e_i) is a node in p with $n_1 < i < i_0$, then $(d_{i-(n_1+1)}, e_i)$ is also a node in p.

This, in turn, implies that all rows are of the same length and that the vertical matching condition is satisfied, as follows.

We start with row length. Consider the tiling word $t_1^{(0)} \cdots t_{n_1}^{(0)} t_0 \cdots t_{i_0-1}$. We already know by choice of n_1 that $t_0 = t_{n_1+1} = N$. We have to show that

1. $t_{\ell \cdot (n_1+1)} = N$ for $1 < \ell < \frac{i_0-1}{n_1+1}$ and
2. for no other t_i, $t_i = N$.

For Point 1, we concentrate on $t_{2(n_1+1)}$, the same argument can be applied inductively for $\ell > 2$. The argument is in fact easy based on (a). We know that $t_{n_1+1} = N$, thus the unique concept name from C satisfied by (d_{n_1+1}, e_{n_1+1}) is N. It follows from (a) that the unique concept name from C satisfied by $(d_{2(n_1+1)}, e_{n_1+1})$ is also N, and thus the same is true for $(d_{2(n_1+1)}, e_{2(n_1+1)})$. Consequently, $t_{2(n_1+1)} = N$. The proof of Point 2 is similar, using (b) instead of (a) and showing that if $t_i = N$ for some t_i not covered by Point 1, then $t_i = N$ for some $i \in \{1, \ldots, n_1\}$ which we know is not the case.

Now for the vertical matching condition. Take any $t_i \neq N$ from $t_1^{(0)} \cdots t_{n_1}^{(0)} t_0 \cdots t_{i_0-1}$ that is neither on the bottommost nor on the topmost row. We know that (d_i, e_i) satisfies a unique concept name $B_{t_i}^j$ from C. By (a), (d_{i+n_1+1}, e_i) is a node on p. It must clearly also satisfy $B_{t_i}^j$ and no other concept name from C, and the same is true for d_{i+n_1+1}. By construction of the tree below a_1, d_{i+n_1+1} satisfies, apart from $B_{t_i}^j$, also a concept name $B_{t'}^{j\oplus 1}$ with $(t_i, t') \in V$. Using the construction of the subtree below a_1 and a_2 and the fact that the prefix p' of p has only $\downarrow\downarrow$-successors, it can be seen that $t_{i+n_1+1} = t'$, which is exactly what we had to show.

"only if". Assume that $\mathcal{P}$ has a solution, that is, there is a tiling τ of some $n \times m$-grid, $n, m \geq 1$. By our assumption on $\mathcal{P}$, we may assume that $m \geq 2$. Let w be a tiling word that is a row by row unfolding of τ, with an additional leading N symbol (that is, every row in w is prefixed by N). The length of w is $(n+1) \cdot m$. We start with showing that $\mathcal{U}$ contains a path p_1 that starts at an S-successor of a_1 and whose labeling with the concept names from C gives rise to w.

The length of p_1 will be $k := (n+1) \cdot (m-1)$; we shall explain later why p_1 is short of one row. We number the columns of the grid from 0 to $n-1$ and the rows of the grid from 0 to $m-1$. For all positions $i \leq k$ on p_1, let

- $\mathsf{row}(i) = (i \operatorname{div}(n+1)) + 1$ and
- $\mathsf{col}(i) = i - 1 \bmod (n+1)$ if $i \bmod (n+1) > 0$ while $\mathsf{col}(i)$ is undefined otherwise.

The '+1' in the first item ensures that the first elements of p_1 corresponds to row 1 rather than to the bottommost row 0. The extra condition in the second items avoids assigning a column to positions in w that carry the symbol N.

By construction of the tree in $\mathcal{U}$ below a_1, we can find a path $p_1 = d_0 \cdots d_k$ that satisfies the following conditions for all $i \leq k$:

1. $(a_1, d_0) \in S^{\mathcal{U}}$;
2. $d_n \in E^{\mathcal{U}}$ (this corresponds to the last position of the first row represented by p_1);
3. if $\tau(\mathsf{col}(i), \mathsf{row}(i)) = t$, then $d_i \in (B_t^{\mathsf{row}(i) \bmod 3})^{\mathcal{U}}$;
4. if $\tau(\mathsf{col}(i), \mathsf{row}(i) - 1) = t$, then $d_i \in (B_t^{\mathsf{row}(i)-1 \bmod 3})^{\mathcal{U}}$;
5. if $\mathsf{col}(i)$ is undefined, then $d_i \in N^{\mathcal{U}}$.

Note that the first $n + 1$ elements on p_1 represent the tiling of row 0 via concept names B_t^0 and the tiling of row 1 via concept names B_t^1. The next $n + 1$ elements represent row 1 via concept names B_t^1 and the tiling of row 2 via concept names B_t^2, and so on.

Once again by construction of the tree below a_1, we can extend p_1 into a path $p_1^+ = d_0 \cdots d_{k+(n+1)}$ that repeats the topmost row $m - 1$ in the sense that the following are satisfied for $k < i \leq k + (n + 1)$:

1. $(d_k, d_{k+1}) \in S^{\mathcal{U}}$;
2. if $\tau(\mathsf{col}(i), m - 1) = t$, then $d_i \in (B_t^{m-1 \bmod 3})^{\mathcal{U}}$;
3. $d_{k+n} \in (B_d^{m-1 \bmod 3})^{\mathcal{U}}$ (this corresponds to the second last position of the repeated row $m - 1$).

So p_1^+ simply repeats the representation of the topmost row from the end of p_1, using the same concept name. The only difference is the labeling with the dummy tile described in Point 3, which is not present in p_1.

By construction of the tree below a_2, $\mathcal{U}$ contains a path $p_2 = e_0 \cdots e_{k+1}$ that starts at an S-successor of $a_2 \in E^{\mathcal{U}}$ and satisfies the following conditions for all $i < k$:

1. $(a_2, e_0) \in S^{\mathcal{U}}$;
2. if $\tau(\mathsf{col}(i), \mathsf{row}(i) + 1) = t$, then $e_i \in (B_t^{\mathsf{row}(i)+1 \bmod 3})^{\mathcal{U}}$;
3. if $\mathsf{col}(i)$ is undefined, then $d_i \in N^{\mathcal{U}}$;
4. e_{k+1} is a cycle node.

Note that the labeling in Point 2 of p_2 is exactly the same as the labeling in Point 3 of p_1.

Now consider the following path in $\mathcal{U} \times \mathcal{U}$ that starts at the S-successor (d_0, e_0) of (a_1, a_2):

- first follow p_1^+ and p_2 snychronously:
$$(d_0, e_0) \cdots (d_{k+1}, e_{k+1})$$
- then proceed to follow p_1^+ while remaining stationary in the cycle node at the end of p_2:
$$(d_{k+2}, e_{k+2}) \cdots (d_k - 1, e_{k+1})$$

- then make a single step downwards in p^1_+, reaching the end of this path, while making a single step upwards in p_2:

$$(d_k, e_k)$$

- then synchronously follow both paths backwards, even stepping up to a_2:

$$(d_{k-1}, e_{k-1}) \cdots (d_n, a_2).$$

By what was said above, it can be verified that (i) the end of this path (d_n, a_2) is in $E^{\mathcal{U}\times\mathcal{U}}$ and (ii) every element of the path satisfies a concept name from C. The only slightly subtle point for the latter is the element (d_{k-1}, e_{k+1}), which is the predecessor of (d_k, e_k) on the constructed path. It satisfies a concept name of the form B^j_d and in fact achieving this is the reason for introducing the dummy tile (as no other concept name from C is satisfied by (d_k, e_k)).

However, there is no path in $\mathcal{U}$ that starts at an S-successor of b and satisfies Properties (i) and (ii); in fact, every path that starts at an S-successor of b and whose end is in $E^{\mathcal{U}}$ must pass an element that does not satisfy any concept name in C. Consequently, $(\mathcal{U}, a_1) \otimes (\mathcal{U}, a_2) \not\rightarrow (\mathcal{U}, b)$, and in fact, the identified S-path yields the witnessing CQ. □

4.4 Size of Witness Queries

In this section, we investigate the size of witness queries. We show that for yes-instances of QBE(Horn-$\mathcal{ALC}$, (U)CQ), there is always a witness query of at most double exponential size, and that there are instances where this double exponential size is unavoidable. The following lemma establishes the lower bound.

Lemma 4.18. *There is a family of Horn-$\mathcal{ALC}$ knowledge bases $(\mathcal{T}_n, \mathcal{A}_n)_{n\geq 1}$, sets of examples S^+ and S^-, a signature Σ, and a polynomial $p(n)$ such that, for all $n \geq 1$, $|\mathcal{T}_n \cup \mathcal{A}_n| \leq p(n)$, $(\mathcal{T}_n, \mathcal{A}_n, S^+, S^-, \Sigma) \in$ QBE(Horn-$\mathcal{ALC}$, (U)CQ) and every (U)CQ witnessing this is of size $\Omega(2^{2^n})$.*

The main idea for the the proof is to give Horn-$\mathcal{ALC}$ knowledge bases $(\mathcal{T}_n, \mathcal{A}_n)$ over two individuals a, b such that in $\mathcal{U}_{\mathcal{A}_n,\mathcal{T}_n}$ the trees below a and b are Σ-homomorphically equivalent to $\mathcal{I}_{2^n}$ and $\mathcal{J}_{2^n}$, respectively, where $\mathcal{I}_n, \mathcal{J}_n$ are given by recursive 'definitions' shown in Figure 4.2. It can be shown that $(\mathcal{I}_n, a) \not\rightarrow_\Sigma (\mathcal{J}_n, b)$, but that $(\mathcal{I}', a) \rightarrow_\Sigma (\mathcal{J}_n, b)$ for any connected proper sub-interpretation $\mathcal{I}'$ of $\mathcal{I}_n$ with $a \in \Delta^{\mathcal{I}'}$. Thus, the smallest Σ-(U)CQ distinguishing between a and b in $(\mathcal{T}_n, \mathcal{A}_n)$ is $(\mathcal{I}_{2^n}, a)$ viewed as CQ, whose size is $\Omega(2^{2^n})$.

Proof. Given some $n \geq 1$, we construct a Horn-$\mathcal{ALC}$-KB $(\mathcal{T}_n, \mathcal{A}_n)$ using concept names $A, B, G, H, U, X_1, \overline{X}_1, C_1, \overline{C}_1, \ldots, X_n, \overline{X}_n, C_1, \overline{C}_n$ and a single role name r. The concept

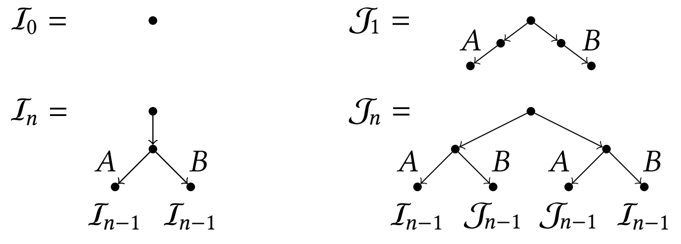

Figure 4.2: The construction shows interpretations such that for every $n \geq 1$, $(\mathcal{I}_n, a) \not\to_\Sigma (\mathcal{J}_n, b)$, but $(\mathcal{I}', a) \to_\Sigma (\mathcal{J}_n, b)$ for any proper sub-interpretation $\mathcal{I}'$ of $\mathcal{I}_n$, where a is the root of $\mathcal{I}_n$ and b is the root of $\mathcal{J}_n$.

names $X_i, \overline{X}_i$ are used to implement an exponential counter. The ABox $\mathcal{A}_n$ is given by

$$\begin{aligned}\mathcal{A}_n = \{&G(a), \overline{X}_1(a), \ldots, \overline{X}_n(a)\} \cup \\ \{&H(b), \overline{X}_1(b), \ldots, \overline{X}_n(b)\}\end{aligned}$$

For the construction of the TBox, we start with including the following CIs:

$$\begin{aligned}
\overline{X}_i &\sqsubseteq U, \quad \text{for all } 1 \leq i \leq n \\
\overline{X}_i &\sqsubseteq U', \quad \text{for all } 1 \leq i < n \\
G \sqcap U &\sqsubseteq \exists r.\,(\exists r.(G \sqcap A) \sqcap \exists r.(G \sqcap B)) \\
H \sqcap U' &\sqsubseteq \exists r.\,(\exists r.(G \sqcap A) \sqcap \exists r.(H \sqcap B)) \sqcap \\
& \exists r.\,(\exists r.(G \sqcap B) \sqcap \exists r.(H \sqcap A)) \\
H \sqcap U &\sqsubseteq \exists r.\exists r.A \sqcap \exists r.\exists r.B
\end{aligned}$$

Note that U and U' are enforced if the counter value is smaller than $2^n - 1$ and $2^n - 2$, respectively. It remains to implement the counter:

$$\begin{aligned}
X_1 &\sqsubseteq \forall r.\forall r.(\overline{X}_1 \sqcap C_1) \\
\overline{X}_1 &\sqsubseteq \forall r.\forall r.(X_1 \sqcap \overline{C}_1) \\
X_i &\sqsubseteq \forall r.\forall r.(\neg C_{i-1} \sqcup (\overline{X}_i \sqcap C_i)), \quad \text{for all } 2 \leq i \leq n \\
\overline{X}_i &\sqsubseteq \forall r.\forall r.(\neg C_{i-1} \sqcup (X_i \sqcap \overline{C}_i)), \quad \text{for all } 2 \leq i \leq n \\
X_i &\sqsubseteq \forall r.\forall r.(\neg \overline{C}_{i-1} \sqcup (X_i \sqcap \overline{C}_i)), \quad \text{for all } 2 \leq i \leq n \\
\overline{X}_i &\sqsubseteq \forall r.\forall r.(\neg \overline{C}_{i-1} \sqcup (\overline{X}_i \sqcap \overline{C}_i)), \quad \text{for all } 2 \leq i \leq n
\end{aligned}$$

Obviously, the size of both $\mathcal{A}_n$ and $\mathcal{T}_n$ is bounded by some polynomial in n. Note that by construction, the connected component of a in $(\mathcal{U}_{\mathcal{A}_n,\mathcal{T}_n}, a)$ (that is, with domain $\{\pi \in \Delta^{\mathcal{U}_{\mathcal{A}_n,\mathcal{T}_n}} \mid \pi \text{ starts with } a\}$) restricted to signature $\Sigma = \{r, A, B\}$ is isomorphic to the interpretation $\mathcal{I}_{2^{n-1}}$ described in Figure 4.2 and the connected component of b in $(\mathcal{U}_{\mathcal{A}_n,\mathcal{T}_n}, b)$ restricted to Σ is isomorphic to $\mathcal{J}_{2^{n-1}}$, for every $n \geq 1$. Let $\mathcal{I}$ denote the subtree of $\mathcal{U}_{\mathcal{A}_n,\mathcal{T}_n}$ starting at a restricted to signature Σ. Define $q_n(x)$ as $(\mathcal{I}, a)$, viewed as CQ. By construction, q_n is a finite binary tree of exponential depth, thus of double exponential size. We verify the following claim:

Claim. $\mathcal{T}_n, \mathcal{A}_n \not\models q_n(b)$, but $\mathcal{T}_n, \mathcal{A}_n \models q'(b)$ for every proper subquery q' of q_n.

Proof of the Claim. For the first part of the claim it suffices to note that, by construction, $(\mathcal{I}_k, a) \not\to_\Sigma (\mathcal{J}_k, b)$ for every $k \geq 1$. For the second part, we show that for every proper sub-interpretation $\mathcal{I}'$ of $\mathcal{I}_k$, there is a homomorphism $(\mathcal{I}', a) \to (\mathcal{J}_k, b)$. We show this by induction on k. The case $k = 1$ is clear, so let $k > 1$ and we assume the statement is true for $k - 1$. We distinguish cases on which assertion is missing in $\mathcal{I}'$.

- If an atom is removed inside one of the two subtrees that are copies of $\mathcal{I}_{k-1}$ that were used to construct $\mathcal{I}_k$, then by induction hypothesis, this subtree can be homomorphically mapped to $\mathcal{J}_{k-1}$ and it is easy to see that $(\mathcal{I}', a) \to (\mathcal{J}_k, b)$.
- If one of the three r-atoms (rooted at a or at the r-successor of a) is removed, it is also easy to see that $(\mathcal{I}', a) \to (\mathcal{J}_k, b)$.
- If the A-atom (resp. the B-atom) at depth 2 is removed, then $\mathcal{I}'$ has a homomorphism into itself which maps the element of the removed A-atom (resp. B-atom) to the other element at depth 2, so $\mathcal{I}'$ is homomorphically equivalent to the interpretation where the root has an outgoing r-path of length 2 to an element that is labelled with B (resp. A) and where there is a copy of $\mathcal{I}_{k-1}$ is rooted at that element. The latter interpretation clearly maps into $\mathcal{J}_k$, which implies $(\mathcal{I}', a) \to (\mathcal{J}_k, b)$.

This finishes the proof of the claim. Now we set $S^+ = \{a\}$ and $S^- = \{b\}$, which finishes the construction of the instance claimed in the lemma. In particular, the claim shows that every UCQ q of size smaller $|\mathcal{I}_{2^{n-1}}|$ with $\mathcal{T}_n, \mathcal{A}_n \models q(a)$ also satisfies $\mathcal{T}_n, \mathcal{A}_n \models q(b)$, so no such query can be a witness for the constructed instance. $\square$

The following lemma establishes the matching upper bound.

Lemma 4.19. *If* $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma) \in$ QBE(Horn-$\mathcal{ALC}$, (U)CQ)*, then there is a witness query of at most double exponential size.*

Proof. We focus on the CQ case, the UCQ case is similar. Let $\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma$ be given. Recall from Section 4.3 that $(\mathcal{P}, \mathbf{a}^*)$ denotes the sub-interpretation of $(\mathcal{U}^k_{\mathcal{A},\mathcal{T}}, \mathbf{a}^*)$ restricted to elements reachable from $N = \mathsf{ind}(\mathcal{A})^k$. We prove that if there is a Σ-homomorphism from an exponentially deep initial part of $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ for some $\mathbf{b} \in S^-$, then there is also a Σ-homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, and thus by Lemmy 4.8 also $(\mathcal{U}^k_{\mathcal{A},\mathcal{T}}, \mathbf{a}^*) \to_\Sigma (\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$. Conversely, this means that if there is no Σ-homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ for any $\mathbf{b} \in S^-$, then an exponentially deep (and thus, doubly exponentially large) initial piece of $\mathcal{P}$ is a witness CQ.

Remember that $\mathcal{P}$ is pseudo tree-shaped with core N and define $\widehat{\mathcal{P}}$ to be $\mathcal{P}$ restricted to the core and all individuals in the trees of $\mathcal{P}$ that have distance at most $|\mathsf{TP}| \cdot (|\mathsf{ind}(\mathcal{A})| + |\mathsf{TP}|) + 1$ from the core. Assume there is a homomorphism h from $(\widehat{\mathcal{P}}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ for some $\mathbf{b} \in S^-$. By the pigeonhole principle, on every path $\mathbf{p}_1 \dots \mathbf{p}_n$ of length $|\mathsf{TP}| \cdot$

$(|\mathsf{ind}(\mathcal{A})|+|\mathsf{TP}|)+1$ in a tree of $\widehat{\mathcal{P}}$ starting from the root of that tree going only downwards, there must be at least one $\mathbf{p}_j$ such that there is a $\mathbf{p}_i$ with $i < j$ and

$$\mathsf{tail}(\mathbf{p}_i) = \mathsf{tail}(\mathbf{p}_j) \text{ and } \mathsf{tail}(h(\mathbf{p}_i)) = \mathsf{tail}(h(\mathbf{p}_j)), \qquad (*)$$

since there are only $|\mathsf{TP}|$ many different possibilities for $\mathsf{tail}(\mathbf{p}_j)$ and only $|\mathsf{ind}(\mathcal{A})| + |\mathsf{TP}|$ many different possibilities for $\mathsf{tail}(h(\mathbf{p}_j))$. Let $D \subseteq \Delta^{\widehat{\mathcal{P}}}$ be the set that contains the first such $\mathbf{p}_j$ on every path, that is, for every such path $\mathbf{p}_1 \dots \mathbf{p}_n$, D contains the element $\mathbf{p}_j$ such that j is the smallest number such that there exists a $i < j$ with $(*)$.

We inductively define a homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ via an infinite sequence $h_0, h_1, \dots$ of homomorphisms with increasing domains. In the following, when we speak of a leaf in $\mathsf{dom}(h_\ell)$, we mean an element of $\mathsf{dom}(h_\ell) \setminus N$ such that no successor of that element is in $\mathsf{dom}(h_\ell)$. While defining the h_ℓ, we maintain the following induction hypothesis:

(IH) h_ℓ is a partial Σ-homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$ whose domain is connected, includes N and for every leaf $\mathbf{p}_j$ in $\mathsf{dom}(h_\ell)$, there is a $\mathbf{p}_i$ on the path from N to $\mathbf{p}_j$ such that $(*)$ holds for h_ℓ. Furthermore, for every $\mathbf{p} \in \mathsf{dom}(h_\ell)$ that is not a leaf in $\mathsf{dom}(h_\ell)$, h_ℓ is defined for all successors of $\mathbf{p}$.

Let h_0 be the restriction of h such that $\mathsf{dom}(h_0)$ is the smallest connected set that includes N and D, which means that all elements in D are leaves in $\mathsf{dom}(h_0)$. Clearly, (IH) holds for h_0.

If h_ℓ is already constructed, we construct $h_{\ell+1}$ by extending h_ℓ to all successors of leaves of $\mathsf{dom}(h_\ell)$, as follows. For every leaf $\mathbf{p} \in \mathsf{dom}(h_\ell)$, let $\mathbf{c} = \mathsf{tail}(\mathbf{p})$ and let $\mathbf{p}' \in \mathsf{dom}(h_\ell)$ the tuple on the path from N to $\mathbf{p}$ that fulfils $(*)$, as guaranteed by (IH), and let $\mathbf{c}' = \mathsf{tail}(\mathbf{p}')$. For every successor of $\mathbf{p}$, introduced via $\mathbf{c} \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$ for some role name r and some tuple $\mathbf{d}$, there is also a r-successor of $\mathbf{p}'$, introduced via $\mathbf{c}' \hookrightarrow_r^{\mathcal{T},\mathcal{A}} \mathbf{d}$, since $\mathbf{c} = \mathbf{c}'$. By (IH), h_ℓ is defined for all successors of $\mathbf{p}'$, in particular, h_ℓ is defined for $\mathbf{p}'r\mathbf{d}$. If $h_\ell(\mathbf{p}'r\mathbf{d}) \in \mathsf{ind}(\mathcal{A})$, we set $h_{\ell+1}(\mathbf{p}r\mathbf{d}) = h_\ell(\mathbf{p}'r\mathbf{d})$. Otherwise, $h_\ell(\mathbf{p}'r\mathbf{d})$ lies in the anonymous part of $\mathcal{U}_{\mathcal{A},\mathcal{T}}$, say $h_\ell(\mathbf{p}'r\mathbf{d}) = h_\ell(\mathbf{p}')rt$ for some $t \in \mathsf{TP}$, and we set $h_{\ell+1}(\mathbf{p}r\mathbf{d}) = h_\ell(\mathbf{p})rt$.

It is now easy to see that $h_{\ell+1}$ is again a Σ-homomorphism that fulfils (IH) and that in the limit, the h_ℓ define a Σ-homomorphism from $(\mathcal{P}, \mathbf{a}^*)$ to $(\mathcal{U}_{\mathcal{A},\mathcal{T}}, \mathbf{b})$, as required. □

4.5 Discussion and Future Work

We have initiated the research on the query-by-example approach for querying in the context of OES. Based on model-theoretical characterizations relying on universal models, we have given foundational theoretical results for QBE($\mathcal{L}$, CQ) and QDEF($\mathcal{L}$, CQ) for $\mathcal{L} \in \{\mathcal{ELI}, \text{Horn-}\mathcal{ALCI}, \text{Horn-}\mathcal{ALC}\}$, and we have determined tight bounds for the size of witness CQs in the worst case. A surprising result is that for Horn-$\mathcal{ALC}$, the extension with inverse roles makes the difference between coNExpTime-completeness and undecidability.

The undecidability result suggests that problems where inverse roles play a role in combination with products of models lead to complicated effects and can easily result in the undecidability of these problems. In [Fun+19], several related problems were proven to be undecidable using the same proof strategy.

Our investigation opens a whole new research avenue towards improving the usability of ontology-enriched systems. From the theoretical perspective, the most natural next step is to broaden the understanding to different ontology and query languages.

Given the state of the art of OES, we are particularly interested in 'lightweight' DLs, such as *DL-Lite* and $\mathcal{EL}$; the model-theoretic characterizations already provide a solid basis for these logics. For non-Horn or Datalog$^{\pm}$ ontologies it will be more challenging – a good starting point for non-Horn DLs might be [Bot+19]. As for the query language, one could also consider regular path queries. From the practical perspective, one can develop systems for QBE over KBs which not only implement reverse-engineering algorithms, but also guide the user in an interactive process to find the desired query, as done in [BCS14; DAB16]. Given the high complexity of QBE, it will be also important to design heuristics [TCP14; Mot+16] or approximations [BR17], as for relational databases. Another possible way to approximate the problem is to bound the size of the witness queries.

The conference paper [GJS18b], on which this chapter is based on, also includes results on the combination QBE(Horn-$\mathcal{ALCI}$, UCQ), which we have not covered in this chapter, but which are included in Table 4.1. We refrained from presenting these results here, since obtaining these results is rather technical and involved, while the query-by-example problem for UCQs is also less interesting from a practical point of view. From a theoretical perspective, however, this problem seems very natural, since by Theorem 4.5, it comes down to checking the existence of a Σ-homomorphism between two universal models of knowledge bases. Another unsolved problem concerns the sizes of witness queries in QBE(Horn-$\mathcal{ALCI}$, UCQ). In [GJS18b], we give a fourfold-exponential upper bound, but it is unclear whether a triple-exponential or double-exponential upper bound can be established, or if there are cases that require such large witnesses.

Related within DL research is the study of *query conservative extensions (QCE)*, where the question is whether two given ontologies or two knowledge bases can be distinguished by a query (without providing examples). Indeed, in the context of QCE, characterizations based on homomorphisms and universal models have been devised and inverse roles also tend to increase the complexity, see [Bot+16] for a recent survey, and references therein. We are, however, not aware of any direct reductions between QBE and QCE.

Within the broader context of machine learning, we believe that the results in this chapter lay the foundations for questions related to *learnability* of queries, see [CP95] for an overview. In this line, one could investigate an ILP inspired variant: if an instance $(\mathcal{T}, \mathcal{A}, S^+, S^-, \Sigma)$ of QBE does *not* have a witness, is there an extension $\mathcal{T}' \supseteq \mathcal{T}$ such that there is a witness? In the context of active learning, one would be interested in learning a (conjunctive) query with membership and/or equivalence queries over a DL knowledge base. Finally, it would be interesting to extend the recently introduced framework of learning concepts over background structures of small degree and having only *local access* to the data [GR17] with an ontology.

5 Query Expressibility and Verification in the Data Integration Setting

In large enterprises that maintain huge amounts of data, it is often the case that the data comes from multiple sources and is stored across several databases using different schemata. Nevertheless, it is crucial for these companies to have the ability to query data efficiently. *Data integration* is a classical approach to this problem, where the data sources are kept untouched and a new, *global schema* is created. One formulates *mappings*, which relate the vocabulary of the global schema to the vocabulary of the original data sources. The global schema then becomes the only intended point of access.

In *Ontology based data access (OBDA)*, one goes one step further. The distinguishing feature here is the presence of an ontology formulated in the global schema, which enables the formulation of OMQs over the global schema and thus, additional query answers can be derived via logical reasoning. Queries over the global schema are then translated back into queries over the data sources and executed in the existing database systems.

In practice, OBDA is often approached in an incremental manner [TLN99; Kha+15; SM17]. One starts with a small set of important source queries (typically hand crafted by experts from the enterprise's IT department) and builds mappings for the involved sources and an initial ontology that support these queries, manually or with the help of extraction tools [Jim+15; Pin+18]. The outcome of this first step is then evaluated and, when considered successful, ontology and mappings are extended to support additional queries. This process may proceed for several rounds and in fact forever since new data sources and queries tend to appear as the enterprise develops and existing data sources or the ontology need to be updated [Lem+17].

When data sources are numerous, data integration is often a considerable investment, since the construction of both the mappings and the ontology is non-trivial and labour intensive. Nevertheless, the OBDA data integration setting is already used in practice. The energy company Equinor ASA (formerly Statoil) uses such a system [Kha+15], and the system Optique is developed with a focus on real time OBDA for data streams.

Reasoning problems

In this chapter, we study two problems that occur in OBDA. The *expressibility problem* asks whether a given source query q_s is already expressible as a target query (that is, over the global schema) and the *verification problem* asks, additionally given a candidate target query q_t, whether q_t expresses q_s. We consider (U)CQs as source and target queries and GAV (global-as-view) mappings, which are mappings that relate a symbol from the global vocabulary to one ore more CQs over the sources. As ontology languages we consider DL-Lite and description logics between $\mathcal{EL}$ and $\mathcal{ELHI}$, which are all very common choices in OBDA. It follows from results in [NSV10; Afr11] that, even without ontologies, additional source UCQs become expressible when full first-order logic (FO) is admitted for the target query rather than only UCQs. In OBDA, however, going beyond UCQs quickly results in undecidability of query answering [Baa+17] and thus we stick with UCQs.

Possible reasons for non-expressibility include that the mappings do not transport all data required for answering q_s to the global schema and that the ontology 'blurs' the distinction between different relations from the sources. If q_s is not expressible, one might decide to add more mappings or to rework the ontology. The following example illustrates this process.

Example 5.1. *Assume the data sources contain a binary relation* Man *with* $\mathsf{Man}(m, d)$ *meaning that the manager m manages the department d and a ternary relation* $\mathsf{Emp}(e, d, o)$ *meaning that employee e works for department d in office o.*[1] *We start with a global schema that consist of one concept name* Employee *and a binary relation* manages*, defined by the following two mappings:*

$$\begin{aligned} \mathsf{Man}(x,z) \land \mathsf{Emp}(y,z,u) &\rightarrow \mathsf{manages}(x,y) \\ \mathsf{Emp}(x,y,z) &\rightarrow \mathsf{Employee}(x) \end{aligned}$$

Then the source query $q_s(x) \leftarrow \mathsf{Man}(x,y)$ *is not expressible because the mappings do not provide sufficient data from the source. In particular, the query* $q_t(x) \leftarrow \mathsf{manages}(x,y)$ *does not express* q_s*, since the source might contain an entry like* $\mathsf{Man}(\mathsf{smith}, \mathsf{legalDepartment})$ *but no entries for employees in the legal department, so that the first mapping does not 'bite'. It trivially becomes expressible as* $q_t(x) \leftarrow \mathsf{Manager}(x)$ *when we add the mapping*

$$\mathsf{Man}(x,y) \rightarrow \mathsf{Manager}(x).$$

Next, we further add the following $\mathcal{EL}$-ontology $\mathcal{T}$:

$$\begin{aligned} \mathsf{Manager} &\sqsubseteq \mathsf{Employee} \\ \mathsf{Manager} &\sqsubseteq \exists \mathsf{manages}.\mathsf{Secretary} \end{aligned}$$

Then the source query $q_s(x) \leftarrow \mathsf{Emp}(x,y,z)$*, which formerly was expressible as* $q_t(x) \leftarrow \mathsf{Employee}(x)$*, is no longer expressible due to the first CI in $\mathcal{T}$. Informally, all the required data is there, but it is mixed with other data and we have no way to separate. The source query* $q_s(x,y) \leftarrow \mathsf{Man}(x,z) \land \mathsf{Emp}(y,z,u)$*, however, is expressible as* $q_t(x,y) \leftarrow \mathsf{manages}(x,y)$ *despite the second CI in $\mathcal{T}$, intuitively because the additional data mixed into* manages *by that CI always involves an anonymous constant introduced through the existential quantifier and is thus never returned as a certain answer.*

[1]Note that relations in the data sources can have large arity, but the arity of symbols in the global schema is at most binary, since it is a DL schema.

The expressibility problem is interesting especially when new queries should be implemented into the system. The verification problem is useful for example when a complex query q_t has been manually constructed to express q_s and when the ontology, mappings, or source schemas have been updated, with an unclear impact on whether q_t still expresses q_s.

Related Work

The expressibility problem and the verification problem have been considered in the context of open data publishing, there called finding and recognition of source-to-target rewritings [Cim17]. There, the general framework is described and algorithms for the the two problems are given for the case of DL-Lite ontologies.

Besides this, the expressibility problem in OBDA is closely related to the problem of query expressibility over views, which has been intensively studied in database theory, see for example [Lev+95; DG97; Cal+02; NSV10; Afr11] and references therein. The problem has occasionally also been considered in a DL context [CDL00; HM05; BLR97; Cal+12]. These papers, however, study setups different from the one we consider, both regarding the rôle of the ontology and the description logics used.

Contribution and Structure of the Chapter

Our main results are that within the setup described above, expressibility and verification are Π_2^p-complete in DL-Lite$^{\mathcal{R}}_{\text{horn}}$ and in many other dialects of DL-Lite, coNExpTime-complete in DLs between $\mathcal{EL}$ and $\mathcal{ELHI}$ when the source UCQ is rooted, and 2-ExpTime-complete in the unrestricted case. There are some surprises here. First, the Π_2^p lower bound already applies when the ontology is empty and the source query is a CQ which means that, in the database theory setting, it is Π_2^p-hard to decide the fundamental problem whether a source CQ is expressible as a (U)CQ over a set of UCQ views. For this problem, an NP upper bound was claimed without proof in [Lev+95], but our results show that the problem is actually Π_2^p-complete. A second surprise is that 2-ExpTime- respectively coNExpTime-hardness applies already in the case where the ontology is formulated in $\mathcal{EL}$ (and when queries are UCQs). We are not aware of any other reasoning problem for $\mathcal{EL}$ that has such a high complexity whereas there are several such problems known for $\mathcal{ELI}$ [Bie+16].

In Section 5.1, we introduce the chapter-specific notions and formally define the two reasoning problems. In Section 5.2, we prove a characterization for expressibility that we use throughout the rest of the chapter. In Section 5.3, we prove Π_2^p-completeness for DL-Lite. The upper bounds for expressibility in $\mathcal{ELHI}$ are proved in Section 5.4 for rooted source queries and in Section 5.5 for unrestricted source queries. The upper bounds for the verification problem in $\mathcal{ELHI}$ are proved in Section 5.6. In Section 5.7, we prove the matching lower bounds for expressibility and verification problems in $\mathcal{EL}$ and we give a conclusion in Section 5.8.

5.1 Preliminaries

We introduce databases, GAV mappings, OBDA specifications, OBDA languages and formally define the expressibility problem and the verification problem.

Since OBDA combines the field of description logics with classical database theory, we are facing a clash of notation from both areas in this chapter concerning databases and ABoxes. We introduce databases while briefly describing how they relate to ABoxes and what vocabulary we use.

Let $\mathbf{S}$ be a signature. A *fact over* $\mathbf{S}$ is an expression of the form $R(a_1 \ldots a_n)$, where $R \in \mathbf{S}$ is an n-ary symbol and $a_i \in \mathsf{N_I}$. A $\mathbf{S}$-*database* is a finite non-empty set D of facts over $\mathbf{S}$. Thus, the syntactical difference between an ABox and a database is that in an ABox we only have relations of arity one and two, while in a database, any arity is allowed. While we formally defined the a_i to come from the set of individuals, when they appear in databases they are traditionally called *constants*. The *active domain of a database* D, denoted $\mathsf{adom}(D)$, is the set of all constants that appear in some fact in D, so adom is the analogue of ind for ABoxes.

To avoid dealing with special cases, we work with a slightly more general definition of CQs in this chapter. For a conjunctive query $q(\mathbf{x}) \leftarrow R_1(\mathbf{y}_1) \wedge \ldots \wedge R_n(\mathbf{y}_n)$ over the schema $\mathbf{S}$, we do *not* require that all answer variables actually occur in some $\mathbf{y}_i$. All further definitions for CQs and UCQs introduced in Section 2.4 are now to be understood with this version of CQs. In particular, $\mathsf{var}(q)$ is the set of all variables that occur in q, including answer variables that do not occur in the body of q, a UCQ is now a set of such CQs, where as usual, every disjunct has the same arity, and a homomorphism from a CQ q to a database D has to be defined also for answer variables that do not occur in the body (but there are no restrictions put on where these variables are mapped to).

Let $q_1(\mathbf{x}_1), q_2(\mathbf{x}_2)$ be UCQs of the same arity and over the same schema $\mathbf{S}$. We say that q_1 is *contained* in q_2, denoted $q_1 \subseteq_{\mathbf{S}} q_2$, if for every $\mathbf{S}$-database D, $\mathsf{ans}_{q_1}(D) \subseteq \mathsf{ans}_{q_2}(D)$. It is well-known that, when q_1 and q_2 are CQs, then $q_1 \subseteq_{\mathbf{S}} q_2$ if and only if there is a homomorphism from q_2 to q_1, that is, a function $h : \mathsf{var}(q_2) \longrightarrow \mathsf{var}(q_1)$ such that $R(h(\mathbf{x})) \in q_1$ for every relational atom $R(\mathbf{x}) \in q_2$ and $h(\mathbf{x}_2) = \mathbf{x}_1$. We indicate the existence of such a homomorphism with $q_2 \longrightarrow q_1$. When q_1 and q_2 are UCQs, then $q_1 \subseteq_{\mathbf{S}} q_2$ if and only if for every disjunct $p_1 \in q_1$ there is a disjunct $p_2 \in q_2$ such that $p_2 \longrightarrow p_1$.

We shall frequently view CQs as databases whose facts are the body of the CQ, just using variables as individuals. Conversely, we shall also view a tuple $(D, \mathbf{a})$ with D a database and $\mathbf{a} = a_1 \cdots a_n \in \mathsf{adom}(D)^n$ as an n-ary CQ; note that repeated elements are admitted in $\mathbf{a}$. We do this by using the facts of D as the body of q (now using individuals as variables) and adding the head $q(\mathbf{a})$.

A *global as view (GAV) mapping* over a schema $\mathbf{S}$ takes the form $\varphi(\mathbf{x}, \mathbf{y}) \longrightarrow \psi(\mathbf{x})$ where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of relational atoms over $\mathbf{S}$ and $\psi(\mathbf{x})$ is of the form $A(x)$, $r(x, y)$, or $r(x, x)$ with A a concept name and r a role name. We call $\varphi(\mathbf{x}, \mathbf{y})$ the *body* of the mapping and $\psi(\mathbf{x})$ its *head*. Every variable that occurs in the head must also occur in the body, so a single GAV mapping corresponds to a unary or binary CQ over the source schema. Let $\mathbf{M}$ be a set of GAV mappings over a schema $\mathbf{S}$. For every $\mathbf{S}$-database D, the mappings in

$\mathbf{M}$ produce an ABox $\mathbf{M}(D)$, defined as follows:

$$\mathbf{M}(D) = \{R(\mathbf{a}) \mid D \models \varphi(\mathbf{a}, \mathbf{b}) \text{ and } \varphi(\mathbf{x}, \mathbf{y}) \to R(\mathbf{x}) \in \mathbf{M}\}.$$

This ABox can be physically materialized or left virtual; we do not make any assumptions regarding this issue. Also note that there can be several mappings with the same head in $\mathbf{M}$, which means that the concept names and roles are de facto defined in terms of UCQs over the source schema.

An *OBDA specification* is a triple $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ where $\mathbf{S}$ is the *source schema*, $\mathbf{M}$ a finite set of mappings over $\mathbf{S}$, and $\mathcal{T}$ a TBox.[2] By $\mathsf{sch}(\mathbf{M})$ we denote the schema that consists of all relation names that occur in the heads of mappings in $\mathbf{M}$. Informally, $\mathcal{S}$ is addressing source data in schema $\mathbf{S}$, translated into an $\mathsf{sch}(\mathbf{M})$-ABox via mappings from $\mathbf{M}$ and then evaluated under the ontology $\mathcal{T}$. Note that $\mathcal{T}$ can use the relation names in $\mathsf{sch}(\mathbf{M})$ as well as additional concept and role names, and so can queries that are posed against the ABox.

We use $[\mathcal{L}, \mathcal{M}]$ to denote the set of all OBDA specifications $(\mathcal{T}, \mathbf{M}, \mathbf{S})$ where $\mathcal{T}$ is formulated in the ontology language $\mathcal{L}$ and all mappings in $\mathbf{M}$ are formulated in the mapping language $\mathcal{M}$ and call $[\mathcal{L}, \mathcal{M}]$ an *OBDA language*. An example of an OBDA language is $[\mathcal{ELHI}, \text{GAV}]$. In this thesis, we shall concentrate on GAV mappings. While other types of mappings such as LAV and GLAV are also interesting [Pog+08; Cim17], they are outside the scope of this thesis.

Definition 5.2. *Let $\mathcal{Q}_s$ and $\mathcal{Q}_t$ be query languages and $[\mathcal{L}, \mathcal{M}]$ an OBDA language.*

1. *The* $\mathcal{Q}_s$-to-$\mathcal{Q}_t$ verification problem in $[\mathcal{L}, \mathcal{M}]$ *is to decide, given an OBDA specification $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S}) \in [\mathcal{L}, \mathcal{M}]$, a source query $q_s \in \mathcal{Q}_s$, and a target query $q_t \in \mathcal{Q}_t$ of the same arity, whether q_t is a* realization of q_s in $\mathcal{S}$, *that is, whether* $\mathsf{ans}_{q_s}(D) = \mathsf{cert}_Q(\mathbf{M}(D))$ *for all* $\mathbf{S}$*-databases D, where $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$.*

2. *The* $\mathcal{Q}_s$-to-$\mathcal{Q}_t$ expressibility problem in $[\mathcal{L}, \mathcal{M}]$ *is to decide, given an OBDA specification $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S}) \in [\mathcal{L}, \mathcal{M}]$ and a source query $q_s \in \mathcal{Q}_s$, whether there is a realization q_t of q_s in $\mathcal{Q}_t$. In the positive case, we say that q_s is* $\mathcal{Q}_t$-expressible in $\mathcal{S}$.

Note that we quantify over all $\mathbf{S}$-databases and not only those where $D \cup \mathcal{T}$ is satisfiable. This however does not make a difference for most setups studied in this chapter, since $\mathcal{EL}$ and $\mathcal{ELHI}$ cannot express inconsistency.

Table 5.1 gives an overview over all complexity results obtained in this chapter.

5.2 Characterizations and Basic Observations

In many classical cases of query expressibility over views, informally stated, q_s is expressible over a set of mappings $\mathbf{M}$ (representing views) if and only if the natural candidate $\mathbf{M}(q_s)$ is a realization of q_s, where $\mathbf{M}(q_s)$ is the UCQ obtained from q_s (seen as a databse)

[2]For readability, we consider a single data source, only. Multiple source databases can be represented as a single one by assuming that their schemas are disjoint and taking the union.

	rUCQ-to-*UCQ*	*UCQ*-to-*UCQ*	*CQ*-to-*CQ*
empty ontology and DL-Lite (*)	Π_2^p-comp.	Π_2^p-comp.	Π_2^p-comp.
between $\mathcal{EL}$ and $\mathcal{ELHI}$	coNExpTime-comp.	2-ExpTime-comp.	in 2-ExpTime

Table 5.1: The table shows the complexity of both the expressibility problem and the verification problem under GAV mappings. (*) The result holds for various dialects of DL-Lite, namely all dialects that fulfil the conditions listed in Theorem 5.12.

by applying the mappings. Furthermore, a query q_t is a realization for q_s if and only if $\mathbf{M}^-(q_t) \equiv q_s$, where $\mathbf{M}^-(q_t)$ is the UCQ obtained from q_t by applying the mappings 'backwards' [NSV10; Afr11]. Our starting point for proving decidability and upper complexity bounds for expressibility in OBDA are the following two observations: First, if q_s is expressible, then $\mathbf{M}(q_s)$ is a realization of q_s, so $\mathbf{M}(q_s)$ is still the natural candidate for a realization, even in the presence of an ontology. Second, to see whether q_s is expressible, we need to check whether $\mathbf{M}^-(q_r)$ is contained in q_s where q_r is a (potentially infinitary) UCQ-rewriting of the UCQ $\mathbf{M}(q_s)$ under the ontology. Verification can be characterized in a very similar way. These characterizations also show that expressibility can be reduced to verification in polynomial time and that if q_s is expressible, then it is expressed by the polynomial size UCQ $\mathbf{M}(q_s)$.

Applying Mappings to Queries, Forwards and Backwards

Let $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ be an OBDA specification and $\mathcal{A}$ an ABox that uses only concept and role names from $\mathsf{sch}(\mathbf{M})$. We say that a mapping $\varphi(\mathbf{x}, \mathbf{y}) \to \psi(\mathbf{x})$ from $\mathbf{M}$ is *suitable* for a fact $\alpha \in \mathcal{A}$ if $\psi(\mathbf{x})$ and α are unifiable, that is, if there exists a function $\sigma : \mathbf{x} \cup \mathbf{y} \to \mathsf{N_I}$ such that $\psi(\sigma(\mathbf{x})) = \alpha$.

We write $\mathbf{M}^-(\mathcal{A})$ to denote the set of S-databases D obtained from $\mathcal{A}$ as follows: for every fact $\alpha \in \mathcal{A}$, choose a suitable mapping $\varphi(\mathbf{x}, \mathbf{y}) \to \psi(\mathbf{x})$ from $\mathbf{M}$ and include $R(\sigma(\mathbf{z}))$ in D whenever $R(\mathbf{z})$ is an atom in $\varphi(\mathbf{x}, \mathbf{y})$ and where $\sigma : \mathbf{x} \cup \mathbf{y} \to \mathsf{N_I}$ is chosen such that $\psi(\sigma(\mathbf{x})) = \alpha$ and every variable from $\mathbf{y}$ is mapped to a fresh constant. Note that $\mathbf{M}^-(\mathcal{A})$ can contain many databases, one for every set of choices of suitable mappings for the facts in $\mathcal{A}$.

Example 5.3. *Let $\mathcal{A} = \{r(a, a), A(a)\}$ and* $\mathbf{M}$ *contains the four mappings*

$$\begin{array}{rcll} S(x,y) & \to & r(x,y) & \quad (1)\\ R(x,y,z) & \to & r(x,y) & \quad (2)\\ S(x,y) & \to & A(x) & \quad (3)\\ T(x) & \to & A(x) & \quad (4)\end{array}$$

There are two suitable mappings for each fact in $\mathcal{A}$, which gives $2 \cdot 2 = 4$ databases in $\mathbf{M}^-(\mathcal{A})$. Choosing mappings (1) and (3) results in $D_{13} = \{S(a, a), S(a, b)\}$. Choosing *mappings (1) and (4) results in $D_{14} = \{S(a,a), T(a)\}$. Choosing mappings (2) and (3) results in $D_{23} = \{R(a,a,b), S(a,c)\}$ and choosing mappings (2) and (4) results in $D_{24} = \{R(a,a,b), T(a)\}$. Thus, $\mathbf{M}^-(\mathcal{A}) = \{D_{13}, D_{14}, D_{23}, D_{24}\}$.*

Both $\mathbf{M}$ and $\mathbf{M}^-$ lift to sets of databases and ABoxes as expected, that is, if S is a set of $\mathbf{S}$-databases, then $\mathbf{M}(S) = \{\mathbf{M}(D) \mid D \in S\}$ and if S is a set of ABoxes over $\mathsf{sch}(\mathbf{M})$, then $\mathbf{M}^-(S) = \bigcup_{\mathcal{A}\in S} \mathbf{M}^-(\mathcal{A})$.

In what follows, we shall often apply $\mathbf{M}$ to a CQ $q(\mathbf{x})$ viewed as a database, and view the result (which formally is an ABox) again as a CQ. In this case, the tuple of answer variables is again $\mathbf{x}$. Notice that it might be the case that not all variables from $\mathbf{x}$ appear in the body of $\mathbf{M}(q(\mathbf{x}))$, so in general, $\mathbf{M}(q(\mathbf{x}))$ is an what we called an unsafe CQ, see Section 2.4. The same applies to UCQs and sets of databases, and to $\mathbf{M}^-(q)$. Note that $\mathbf{M}^-(q)$ gives a UCQ even when q was a CQ, as seen in Example 5.3.

Example 5.4. *Consider the CQ $q(x,x,y) \leftarrow R(x,x) \land S(x,y) \land S(y,z)$ and let $\mathbf{M}$ consist of the single mapping $R(x,y) \rightarrow r(x,y)$, that is, the relation R is simply copied and the relation S is dropped. Then $\mathbf{M}(q)$ viewed as a CQ is $p(x,x,y) \leftarrow r(x,x)$, where the answer variable y does not occur in the body, so $\mathbf{M}(q)$ is unsafe.*

The following fundamental lemma describes the (non)-effect of applying $\mathbf{M}$ and $\mathbf{M}^-$ on query containment. It is explicit or implicit in many papers concerned with query rewriting under views or with query determinacy, see for example [NSV10; Afr11].

Lemma 5.5. *Let $\mathbf{M}$ be a set of GAV mappings, q, q_1 and q_2 UCQs over $\mathbf{S}$ and r, r_1 and r_2 UCQs over $\mathsf{sch}(\mathbf{M})$. Then:*

1. *If $q_1 \subseteq_{\mathbf{S}} q_2$, then $\mathbf{M}(q_1) \subseteq_{\mathsf{sch}(\mathbf{M})} \mathbf{M}(q_2)$.*
2. *If $r_1 \subseteq_{\mathsf{sch}(\mathbf{M})} r_2$, then $\mathbf{M}^-(r_1) \subseteq_{\mathbf{S}} \mathbf{M}^-(r_2)$.*
3. *$q \subseteq_{\mathbf{S}} \mathbf{M}^-(r)$ if and only if $\mathbf{M}(q) \subseteq_{\mathsf{sch}(\mathbf{M})} r$.*

Proof. We prove all statements for CQs, it is straightforward to generalize to UCQs: one only needs to employ the characterization of UCQ in terms of homomorphisms instead of the one for CQs.

1. Since $q_1 \subseteq q_2$, we have $q_2 \rightarrow q_1$. Let h be a homomorphism witnessing this. It can be verified that restricting h to $\mathsf{var}(\mathbf{M}(q_2))$ yields a homomorphism from $\mathbf{M}(q_2)$ to $\mathbf{M}(q_1)$, thus $\mathbf{M}(q_1) \subseteq_{\mathsf{sch}(\mathbf{M})} \mathbf{M}(q_2)$. In fact, let $R(\mathbf{x})$ be a relational atom in $\mathbf{M}(q_2)$. Then $R(\mathbf{x})$ was produced by some mapping $\varphi(\mathbf{y}) \rightarrow R(\mathbf{z}) \in \mathbf{M}$ and homomorphism h' from $\varphi(\mathbf{y})$ to q_2 with $h'(\mathbf{z}) = \mathbf{x}$. Composing h' with h enables an application of the same mapping in q_1 that delivers $R(h(\mathbf{x})) \in \mathbf{M}(q_1)$, as required. Furthermore, since h maps the tuple of answer variables of q_2 to the tuple of answer variables of q_1, so does the restriction of h to $\mathsf{var}(\mathbf{M}(q_2))$.

2. From $r_1 \subseteq_{\mathsf{sch}(\mathbf{M})} r_2$, we obtain $r_2 \longrightarrow r_1$. Let h be a witnessing homomorphism. We need to show that for each disjunct p_1 in the UCQ $\mathbf{M}^-(r_1)$, there is a disjunct p_2 in the UCQ $\mathbf{M}^-(r_2)$ such that $p_2 \longrightarrow p_1$. Thus let p_1 be from $\mathbf{M}^-(r_1)$. By construction of $\mathbf{M}^-(r_1)$, p_1 is obtained from r_1 by choosing a suitable mapping from $\mathbf{M}$ for each relational atom in r_1 and 'applying it backwards'. We identify p_2 by choosing for every atom $R(\mathbf{y})$ of r_2 the mapping chosen for $R(h(\mathbf{y})) \in r_1$ in the construction of p_1. We can then straightforwardly define a homomorphism h' from p_2 to p_1 by extending h to the fresh variables in p_2.

3. For the "only if" direction, assume $p \longrightarrow q$ for some disjunct p in the UCQ $\mathbf{M}^-(r)$ and let h be a witnessing homomorphism. Let h' be the restriction of h to $\mathsf{var}(r)$. It can be verified that h' is a homomorphism from r to $\mathbf{M}(q)$. In fact, let $R(\mathbf{x})$ be a relational atom in r. Then by construction of $\mathbf{M}^-(r)$ there is a mapping $\varphi(\mathbf{y}) \longrightarrow R(\mathbf{z}) \in \mathbf{M}$ that was 'applied backwards' in the construction of p and thus there is a homomorphism g from $\varphi(\mathbf{y})$ to $\mathbf{M}^-(r)$ with $g(\mathbf{z}) = \mathbf{x}$. Composing g with h enables an application of the same mapping in q that delivers $R(h(\mathbf{x})) \in \mathbf{M}(q)$, as required.

   For the "if" direction, assume that there is a homomorphism h from r to $\mathbf{M}(q)$. We need to show that there is a disjunct p in the UCQ $\mathbf{M}^-(r)$ such that $p \longrightarrow q$. By construction, every atom $R(\mathbf{x})$ in $\mathbf{M}(q)$ is produced by some mapping $\varphi(\mathbf{y}) \longrightarrow R(\mathbf{z}) \in \mathbf{M}$ and homomorphism g from $\varphi(\mathbf{y})$ to q with $g(\mathbf{z}) = \mathbf{x}$. We identify p by choosing for every atom $R(\mathbf{y})$ of r the mapping that produced $R(h(\mathbf{y})) \in \mathbf{M}(q)$. We can then straightforwardly define a homomorphism h' from p to q by extending h to the fresh variables in p.

□

Characterization for Verification

The following theorem characterizes realizations in terms of UCQ rewritings and $\mathbf{M}^-$. It can thus serve as a basis for deciding the verification problem.

Theorem 5.6. *Let $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ be an OBDA specification from [FO(=), GAV], q_s a UCQ over* $\mathbf{S}$, *q_t a UCQ over* $\mathsf{sch}(\mathbf{M})$, *and q_r an infinitary UCQ rewriting of the OMQ $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$. Then q_t is a realization of q_s if and only if $q_s \equiv_{\mathbf{S}} \mathbf{M}^-(q_r)$.*

Proof. "if". Assume that $q_s \equiv_{\mathbf{S}} \mathbf{M}^-(q_r)$. We have to show that q_t is a realization of q_s. Since q_r is a rewriting of Q, it suffices to prove that $\mathsf{ans}_{q_s}(D) = \mathsf{ans}_{q_r}(\mathbf{M}(D))$ for all $\mathbf{S}$-databases D.

For "$\subseteq$", assume that $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$. Let p be $(D, \mathbf{a})$ viewed as a CQ. From $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$, we obtain $p \subseteq q_s$, and $q_s \subseteq_{\mathbf{S}} \mathbf{M}^-(q_r)$ yields $p \subseteq_{\mathbf{S}} \mathbf{M}^-(q_r)$. With Point 3 of Lemma 5.5, it follows that $\mathbf{M}(p) \subseteq_{\mathbf{S}} q_r$, which by construction of p implies $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathbf{M}(D))$.

For "$\supseteq$", assume that $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathbf{M}(D))$. Let p be $(\mathbf{M}(D), \mathbf{a})$ viewed as a UCQ. Then, $p \subseteq_{\mathsf{sch}(\mathbf{M})} q_r$ and Point 3 of Lemma 5.5 yields $p' \subseteq_{\mathbf{S}} \mathbf{M}^-(q_r)$ where p' is $(D, \mathbf{a})$ viewed as a CQ. Together with $\mathbf{M}^-(q_r) \subseteq_{\mathbf{S}} q_s$, we obtain $p' \subseteq_{\mathbf{S}} q_s$, which implies that $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$.

For the "only if" direction, assume that q_t is a realization of q_s. We have to show that $q_s \equiv_\mathsf{S} \mathbf{M}^-(q_r)$. Thus, let D be an S-database and $\mathbf{a}$ a tuple from $\mathsf{adom}(D)$ whose length matches the arity of q_s. Further, let p be $(D, \mathbf{a})$ viewed as a database. Since q_t is a realization of q_s and q_r a rewriting of the OMQ Q, $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$ if and only if $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathbf{M}(D))$. The latter is the case if and only if $\mathbf{M}(p) \subseteq_{\mathsf{sch}(\mathbf{M})} q_r$ which by Point 3 of Lemma 5.5 holds if and only if $p \subseteq_\mathsf{S} \mathbf{M}^-(q_r)$. This in turn is the case if and only if $\mathbf{a} \in \mathsf{ans}_{\mathbf{M}^-(q_r)}(D)$. □

Characterization for Expressibility

The next theorem characterizes the expressibility of source queries in an OBDA specification. It has several interesting consequences. First, it implies that the UCQ $\mathbf{M}(q_s)$ is a realization of a UCQ q_s over S if there is any such realization. This is well known in the case without an ontology [NSV10; Afr11] and is implicit in [Cim17] for a rather special case of OBDA. Second, the theorem provides a polynomial time reduction of expressibility to verification: q_s is expressible in $\mathcal{S}$ if and only if $\mathbf{M}(q_s)$ is a realization of q_s in $\mathcal{S}$. And third, it shows that if q_s is a CQ, then CQ-expressibility coincides with UCQ-expressibility. Thus, all lower bounds for CQ-to-CQ expressibility also apply to (U)CQ-to-UCQ expressibility and all upper bounds for UCQ-to-UCQ verification and expressibility also apply to the corresponding CQ-to-(U)CQ case.

To prove the characterization for expressibility, we need the following lemma, which states that an infinitary UCQ rewriting (which is usually evaluated without the ontology) does not give more answers if it is evaluated as an OMQ.

Lemma 5.7. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ with $\mathcal{T}$ formulated in FO(=), q a UCQ, and q_r an infinitary UCQ rewriting of Q. Then $(\mathcal{T}, \Sigma, q_r) \subseteq_\Sigma q_r$.*

Proof. For brevity, let $Q' = (\mathcal{T}, \Sigma, q_r)$. Take a Σ-ABox $\mathcal{A}$ and an $\mathbf{a} \in \mathsf{ind}(\mathcal{A})$ such that $\mathbf{a} \in \mathsf{cert}_{Q'}(\mathcal{A})$. We show that $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$, which implies $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathcal{A})$ as desired since q_r is a rewriting of Q. Let $\mathcal{I}$ be a model of $\mathcal{A}$ and $\mathcal{T}$. We have to show that $\mathbf{a} \in \mathsf{ans}_q(\mathcal{I})$.

From $\mathbf{a} \in \mathsf{cert}_{Q'}(\mathcal{A})$, we obtain $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathcal{I})$. This clearly implies that there is an interpretation $\mathcal{I}_f$ that is obtained by restricting $\mathcal{I}$ to a finite subset of the domain and satisfies $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathcal{I}_f)$. Let $\mathcal{A}_\mathcal{I}$ be $\mathcal{I}_f$ viewed as an ABox, restricted to the symbols in Σ. Since q_r uses only symbols from Σ, $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathcal{A}_\mathcal{I})$. As q_r is a rewriting of Q, $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A}_\mathcal{I})$. Observe that $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}_\mathcal{I}$, and thus it follows that $\mathbf{a} \in \mathsf{ans}_q(\mathcal{I})$, as required. □

We are now ready to state the characterization for expressibility.

Theorem 5.8. *Let $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathsf{S})$ be an OBDA specification from (FO(=), GAV), q_s a UCQ over S, and q_r an infinitary UCQ rewriting of the OMQ $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), \mathbf{M}(q_s))$. Then q_s is UCQ-expressible in $\mathcal{S}$ if and only if $\mathbf{M}^-(q_r) \subseteq_\mathsf{S} q_s$. Moreover, if this is the case, then $\mathbf{M}(q_s)$ is a realization of q_s in $\mathcal{S}$.*

Proof. We first observe that

(a) if $\mathbf{M}^{-}(q_r) \subseteq_{\mathsf{S}} q_s$, then $\mathbf{M}(q_s)$ is a realization of q_s in $\mathcal{S}$.

This actually follows from Theorem 5.6 because $q_s \subseteq_{\mathsf{S}} \mathbf{M}^{-}(q_r)$ always holds. In fact, since $\mathbf{M}(q_s)$ is the actual query in Q and since q_r is a rewriting of Q, we have $\mathbf{M}(q_s) \subseteq_{\mathsf{sch}(\mathbf{M})} q_r$; applying Point 3 of Lemma 5.5 then yields $q_s \subseteq_{\mathsf{S}} \mathbf{M}^{-}(q_r)$.

Note that (a) establishes the "if" part of Theorem 5.8. In view of Theorem 5.6 and by (a), we can prove both the "only if" and the "Moreover" part by showing that if there is any realization q_t of q_s in $\mathcal{S}$, then $\mathbf{M}(q_s)$ is a realization of q_s.

Thus assume that q_t is such a realization and let Q' be the OMQ $(\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$ and q_r' a UCQ-rewriting of Q'. We aim to show that

(b) $\mathbf{M}^{-}(q_r) \subseteq_{\mathsf{S}} \mathbf{M}^{-}(q_r')$.

This suffices since Theorem 5.6 yields $\mathbf{M}^{-}(q_r') \subseteq_{\mathsf{S}} q_s$ and composing (b) with this containment gives $\mathbf{M}^{-}(q_r) \subseteq_{\mathsf{S}} q_s$ that yields the desired result because of (a).

To establish (b), by Point 2 of Lemma 5.5 it suffices to show $q_r \subseteq_{\mathsf{sch}(\mathbf{M})} q_r'$. From Theorem 5.6, we get $q_s \subseteq_{\mathsf{S}} \mathbf{M}^{-}(q_r')$. Point 3 of Lemma 5.5 gives $\mathbf{M}(q_s) \subseteq_{\mathsf{sch}(\mathbf{M})} q_r'$. By the semantics of certain answers, this implies $(\mathcal{T}, \mathsf{sch}(\mathbf{M}), \mathbf{M}(q_s)) \subseteq_{\mathsf{sch}(\mathbf{M})} (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_r')$. Since the former OMQ is just Q and q_r is a rewriting of Q, we get $q_r \subseteq_{\mathsf{sch}(\mathbf{M})} (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_r')$. It thus remains to show $(\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_r') \subseteq q_r'$, which is exactly the statement of Lemma 5.7. □

$\mathbf{M}(q_s)$ is the Strongest Over-Approximation

We have seen that $\mathbf{M}(q_s)$ is a realization of q_s if q_s is expressible. But the natural candidate $\mathbf{M}(q_s)$ has an even stronger property: For a given $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathsf{S}) \in [\mathrm{FO}(=), \mathit{GAV}]$ and q_s a UCQ over S, we call a query q_t over $\mathsf{sch}(\mathbf{M})$ an *over-approximation* if $\mathsf{ans}_{q_s}(D) \subseteq \mathsf{cert}_{Q_t}(\mathbf{M}(D))$ for all S-databases D, where $Q_t = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$. In this section, we show that even if q_s is not realizable, $\mathbf{M}(q_s)$ is still the strongest over-approximation of q_s, that is, it is smallest (regarding OMQ containment) over-approximation of q_s. However, this is under the following condition:

$$\text{For every database } D \text{ and } \mathbf{a} \in \mathsf{ans}_{q_s}(D), \text{ we have } \mathbf{a} \subseteq \mathsf{ind}(\mathbf{M}(D)). \tag{5.1}$$

This condition says that all relevant constants to answer q_s have to be exported. Of course, if this is not the case, then there can not be any OMQ over the global scheme that returns all answers to q_s. It is easy to see that this condition is related to safety of $\mathbf{M}(q_s)$:

Lemma 5.9. *Let $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathsf{S}) \in [FO(=), GAV]$ and q_s a UCQ over S. Then Condition (5.1) holds if and only if $\mathbf{M}(q_s)$ is safe.*

Proof. Assume Condition (5.1) holds. Let $(D, \mathbf{a})$ be $q_s(\mathbf{x})$ seen as a database with the distinguished tuple $\mathbf{a} = \mathbf{x}$. Then $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$ and by Condition (5.1), $\mathbf{a} \subseteq \mathsf{ind}(\mathbf{M}(D))$, so every constant from $\mathbf{a}$ is exported to $\mathbf{M}(D)$ via some mapping from $\mathbf{M}$. This means that every variable from $\mathbf{x}$ appears in the body of $\mathbf{M}(q_s)$, so $\mathbf{M}(q_s)$ is safe.

Now assume that $\mathbf{M}(q_s)$ is safe and let D be a S-database and $\mathbf{a} = a_1 \dots a_n \in \mathsf{ans}_{q_s}(D)$. Then there is a homomorphism h from q_s to D with $h(x_i) = a_i$ for $1 \leq i \leq n$. Since $\mathbf{M}(q_s)$ is safe, every x_i appears in the body of $\mathbf{M}(q_s)$, so for every i there is a homomorphism g_i from the body of a view to q_s with x_i in its range. The composition $h \circ g_i$ yields a homomorphism from the body of a view to D with a_i in its range and thus, $\mathbf{a} \subseteq \mathsf{ind}(\mathbf{M}(D))$. □

The following lemma states that if Condition (5.1) is fulfilled, then $\mathbf{M}(q_s)$ is indeed the strongest over-approximation of q_s.

Lemma 5.10. *Let $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathsf{S}) \in [FO(=), GAV]$ and q_s a* S*-UCQ that fulfils Condition (5.1). Let $Q_s = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), \mathbf{M}(q_s))$ and $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q)$ for some* $\mathsf{sch}(\mathbf{M})$*-UCQ q.*

1. $\mathsf{ans}_{q_s}(D) \subseteq \mathsf{cert}_{Q_s}(\mathbf{M}(D))$ *for all* S*-databases D.*
2. *If* $\mathsf{ans}_{q_s}(D) \subseteq \mathsf{cert}_{Q}(\mathbf{M}(D))$ *for all* S*-databases D, then $Q_s \subseteq_{\mathsf{sch}(\mathbf{M})} Q$.*

Proof. For Point 1, let $\mathbf{a} \in \mathsf{ans}_{q_s}(D)$. Let p be $(D, \mathbf{a})$ seen as a CQ, so we have $p \subseteq_{\mathsf{S}} q_s$. By Condition (5.1), we have $\mathbf{a} \subseteq \mathsf{ind}(\mathbf{M}(D))$. In particular, $\mathbf{a} \in \mathsf{ans}_p(\mathbf{M}(D))$ by the identity homomorphism on $\mathbf{M}(D)$. From Point 1 of Lemma 5.5 it follows that $\mathbf{M}(p) \subseteq_{\mathsf{sch}(\mathbf{M})} \mathbf{M}(q_s)$, so $\mathbf{a} \in \mathsf{ans}_{\mathbf{M}(q_s)}(D)$, which implies $\mathbf{a} \in \mathsf{cert}_{Q_s}(D)$.

For Point 2, assume $\mathsf{ans}_{q_s}(D) \subseteq \mathsf{cert}_Q(\mathbf{M}(D))$ holds for all S-databases D. We show that $Q_s \subseteq_{\mathsf{sch}(\mathbf{M})} Q$. For every disjunct $p(\mathbf{x})$ of q_s, let $(D_p, \mathbf{a}_p)$ be $p(\mathbf{x})$ seen as a database with a distinguished tuple $\mathbf{a}_p = \mathbf{x}$. Of course, we have $\mathbf{a}_p \in \mathsf{ans}_{q_s}(D_p)$ for every such disjunct p, so by assumption, $\mathbf{a}_p \in \mathsf{cert}_Q(\mathbf{M}(D_p))$. Let q_r be a infinitary UCQ rewriting of Q, so $\mathbf{a}_p \in \mathsf{ans}_{q_r}(\mathbf{M}(D_p))$. By definition of D_p, this implies $\mathbf{M}(q_s) \subseteq_{\mathsf{sch}(\mathbf{M})} q_r$. This implies the containment of the OMQs $Q_s \subseteq_{\mathsf{sch}(\mathbf{M})} (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_r)$, where by Lemma 5.7, the latter is contained in q_r, which is equivalent to Q. Thus, $Q_s \subseteq_{\mathsf{sch}(\mathbf{M})} Q$. □

Together, Lemma 5.9 and Lemma 5.10 imply that it can be checked in polynomial time whether there exists a UCQ over-approximating q_s by computing $\mathbf{M}(q_s)$ and checking whether it is safe. Additionally, in the positive case, this already computes the smallest over-approximation, which is just $\mathbf{M}(q_s)$.

The Effect of the Ontology on Expressibility

The following corollary of Theorem 5.8 shows that while making the ontology logically stronger might make some source queries inexpressible (as seen in Example 5.1), it never results in additional such queries becoming expressible. Intuitively this is the case because facts that are newly entailed by the ontology are mixed with facts that were directly produced by a mapping, so it might become unclear what the raw result from the mappings was.

Corollary 5.11. *Let $\mathcal{S}_i = (\mathcal{T}_i, \mathbf{M}, \mathsf{S})$, $i \in \{1, 2\}$, be OBDA specifications from $[FO, GAV]$ with $\mathcal{T}_1 \models \mathcal{T}_2$, $\mathcal{Q} \in \{CQ, UCQ\}$ and q_s from $\mathcal{Q}$. Then $\mathcal{Q}$-expressibility of q_s in $\mathcal{S}_1$ implies $\mathcal{Q}$-expressibility of q_s in $\mathcal{S}_2$.*

Proof. Assume that q_s is $\mathcal{Q}$-expressible in $\mathcal{S}_1$. Then Theorem 5.8 gives that $\mathbf{M}(q_s)$ is a realization, and this query is also from $\mathcal{Q}$. We show that $\mathbf{M}(q_s)$ is also a realization of q_s in $\mathcal{S}_2$. Let $q_{r,i}$ be the canonical infinitary UCQ rewriting of the OMQ $Q_i = (\mathcal{T}_i, \mathsf{sch}(\mathbf{M}), \mathbf{M}(q_s))$, $i \in \{1, 2\}$. By Theorem 5.6, $q_s \equiv_{\mathbf{S}} \mathbf{M}^{-}(q_{r,1})$. Since $\mathcal{T}_1 \models \mathcal{T}_2$, we have $Q_2 \subseteq_{\mathsf{sch}(\mathbf{M})} Q_1$. This clearly implies that every CQ in $q_{r,2}$ is also in $q_{r,1}$. Thus $q_s \equiv_{\mathbf{S}} \mathbf{M}^{-}(q_{r,1})$ implies $q_s \supseteq_{\mathbf{S}} \mathbf{M}^{-}(q_{r,2})$. It remains to argue that $q_s \subseteq_{\mathbf{S}} \mathbf{M}^{-}(q_{r,2})$. Since $q_{r,2}$ is a rewriting of Q_2, we have $Q_2 \subseteq_{\mathsf{sch}(\mathbf{M})} q_{r,2}$. By the semantics and definition of Q_2, $\mathbf{M}(q_s) \subseteq_{\mathsf{sch}(\mathbf{M})} Q_2$ and thus $\mathbf{M}(q_s) \subseteq_{\mathsf{sch}(\mathbf{M})} q_{r,2}$. Point 3 of Lemma 5.5 yields $q_s \subseteq_{\mathbf{S}} \mathbf{M}^{-}(q_{r,2})$ as desired. □

5.3 Expressibility and Verification in DL-Lite

We consider OBDA specifications in which the ontology is formulated in a dialect of DL-Lite. The distinguishing feature of logics from this family is that finite UCQ rewritings of OMQs always exist. Therefore, Theorems 5.6 and 5.8 immediately imply decidability of the verification and expressibility problem, respectively. It is, however, well known that UCQ rewritings can become exponential in size [Got+14] and thus optimal complexity bounds are not immediate.

We consider the dialect DL-Lite$^{\mathcal{R}}_{\text{horn}}$ as a typical representative of the DL-Lite family of logics. However, our results also apply to many other dialects since their proof rests only on the following properties, established in [Art+09].

Theorem 5.12. *In DL-Lite$^{\mathcal{R}}_{\text{horn}}$,*

1. *all OMQs Q have a UCQ-rewriting in which all CQs are of size polynomial in $|Q|$;*
2. *OMQ evaluation is in NP in combined complexity.*

We remark that the results presented in this section are related to those obtained in [Cim17], where the DL-Lite$_{\mathcal{A},id}$ dialect of DL-Lite is considered, mappings are GLAV, and queries CQs. A main difference is that Cima's technical results concern rewritings that are complete but not necessarily sound, which corresponds to replacing '$\mathsf{ans}_{q_s}(D) = \mathsf{cert}_Q(\mathbf{M}(D))$' in Definition 5.2 with '$\mathsf{ans}_{q_s}(D) \subseteq \mathsf{cert}_Q(\mathbf{M}(D))$'. Some of his technical constructions are similar to ours. Note that DL-Lite$_{\mathcal{A},id}$ also satisfies the conditions from Theorem 5.12 and thus our results apply to [DL-Lite$_{\mathcal{A},id}$, GAV] as well.

For an OMQ $Q = (\mathcal{T}, \mathbf{S}, q)$, with $\mathcal{T}$ formulated in FO(=) and q a UCQ, the *canonical UCQ-rewriting of size n* is the UCQ q_c that consists of all pairs $(\mathcal{A}, \mathbf{a})$ viewed as a CQ where $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ and $|\mathcal{A}| \leq n$. The following lemma is interesting in connection with Point 1 of Theorem 5.12 as it allows us to concentrate on canonical UCQ rewritings of polynomial size.

Lemma 5.13. *Let $Q = (\mathcal{T}, \Sigma, q)$ be an OMQ with $\mathcal{T}$ formulated in FO(=) and q a UCQ. If Q has a UCQ-rewriting q_r in which all CQs are of size at most n, then the canonical UCQ-rewriting q_c of size n is also a rewriting of Q.*

Proof. We show that $q_c \equiv_\Sigma q_r$. Since q_r is a UCQ-rewriting of Q, it follows that q_c is also a UCQ-rewriting of Q. We consider the two directions of the equivalence separately.

$q_c \supseteq_\Sigma q_r$. This holds because every CQ q in q_r is also an element in q_c. In fact, $q(\mathbf{x})$ being in q_r means that it is of size at most n. Viewing q as an ABox and $\mathbf{x}$ as a candidate answer, we trivially have $\mathbf{x} \in \mathsf{ans}_{q_r}(q)$ and thus $\mathbf{x} \in \mathsf{cert}_Q(q)$ because q_r is a rewriting of Q. As a consequence, the pair $(q, \mathbf{x})$ gives rise to a CQ in q_c.

$q_c \subseteq_\Sigma q_r$. Let q in q_c. We have to show that there is a CQ q' in q_r such that $q' \to q$. By definition of q_c, q is a pair $(\mathcal{A}, \mathbf{a})$ viewed as a CQ such that $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$. Because q_r is a rewriting of Q, it follows from the latter that $\mathbf{a} \in \mathsf{ans}_{q_r}(\mathcal{A})$. This means that there is some q' in q_r with a homomorphism h from q' to $\mathcal{A}$ that maps the answer variables of q' to $\mathbf{a}$. As q is just $\mathcal{A}$ with $\mathbf{a}$ as the answer variables, h shows $q' \to q$. □

We are now ready to establish the upper bound.

Theorem 5.14. *In [DL-Lite$^{\mathcal{R}}_{\mathsf{horn}}$, GAV], the UCQ-to-UCQ expressibility and verification problems are in Π^p_2.*

Proof. As remarked before Theorem 5.8, expressibility polynomially reduces to verification and thus it suffices to consider the latter. Hence let the following be given: an OBDA specification $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ from [DL-Lite$^{\mathcal{R}}_{\mathsf{horn}}$, GAV], a UCQ q_s over schema $\mathbf{S}$, and a UCQ q_t over the schema $\mathsf{sch}(\mathbf{M})$. Let n be the size of this input.

Let $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$ and note that the size of Q is polynomial in n. By Point 1 of Theorem 5.12, we can assume that Q has a UCQ-rewriting in which all CQs are of size $P(n)$, P a polynomial. By Lemma 5.13, we can even assume that this rewriting is the canonical UCQ-rewriting q_c of size $P(n)$. By Theorem 5.6, q_t is thus a realization of q_s if and only if $q_s \equiv_{\mathbf{S}} \mathbf{M}^-(q_c)$. We show that both inclusions of this equivalence can be checked in Π^p_2.

First, consider the inclusion $q_s \subseteq_{\mathbf{S}} \mathbf{M}^-(q_c)$. It holds if and only if for every q in q_s, there is a p in $\mathbf{M}^-(q_c)$ and a homomorphism $p \to q$. This condition can be checked even in NP: iterate over all CQs q in q_s (of which there are at most n), guess a disjunct p from $\mathbf{M}^-(q_c)$, and verify in NP that $p \to q$.

To guess a p in $\mathbf{M}^-(q_c)$, it suffices to guess a pair $(\mathcal{A}, \mathbf{a})$ in q_c and suitable mappings from $\mathbf{M}$ for every fact in $\mathcal{A}$, which determine p. Then, p can be computed in polynomial time from $\mathcal{A}$ and these suitable mappings. We guess the pair $(\mathcal{A}, \mathbf{a})$ from q_c by guessing an arbitrary ABox $\mathcal{A}$ of size at most $P(n)$ and then verifying that $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$. By Point 2 of Theorem 5.12 this verification is possible in NP.

We next consider the inclusion $\mathbf{M}^-(q_c) \subseteq_{\mathbf{S}} q_s$. This holds if and only if for every p in $\mathbf{M}^-(q_c)$, there is a CQ q in q_s such that $q \to p$. We can thus universally guess a p in $\mathbf{M}^-(q_c)$, then iterate over all CQs q in q_s, and for each such q check in NP whether $q \to p$. For universally guessing p, we actually guess a CQ p of size at most $P'(n)$ and then verify that it is in $\mathbf{M}^-(q_c)$. It has already been argued above that this is possible in NP. Overall, we obtain a Π^p_2-algorithm, as desired. □

We next show that the expressibility problem in [DL-Lite$^{\mathcal{R}}_{\mathsf{horn}}$, GAV] is Π^p_2-hard, and thus the same holds for the verification problem. Interestingly, the lower bound already applies when the ontology is empty and the source query is a CQ. As noted in the introduction to this chapter, this shows that expressibility of a source CQ as a (U)CQ over UCQ views is Π_2^p-hard, and in fact it is Π_2^p-complete by Theorem 5.14. This corrects a (very likely) erroneous statement of NP-completeness in [Lev+95].

Theorem 5.15. *The CQ-to-CQ expressibility problem is Π_2^p-hard for GAV mappings and the empty ontology.*

The proof is by reduction of validity of $\forall\exists$-3SAT formulas. By Theorem 5.8, expressibility in the absence of an ontology amounts to checking the containment $\mathbf{M}^-(\mathbf{M}(q_s)) \subseteq q_s$, which is equivalent to the $\forall\exists$-statement that for all $p \in \mathbf{M}^-(\mathbf{M}(q_s))$ there exists a homomorphism $q_s \to p$. Hence, we encode a $\forall\exists$-3SAT formula such that the outer universal quantifiers correspond to the different choices of mappings when taking a $p \in \mathbf{M}^-(\mathbf{M}(q_s))$, whereas the inner existential quantifiers of the formulas correspond to homomorphisms $q_s \to p$.

In preparation for the proof, let us recall the standard representation of 3SAT as a constraint satisfaction problem (CSP) and sketch how this can be used to show that the CQ-to-CQ expressibility problem is NP-hard for GAV mappings and the empty ontology. Let $\varphi(y_1, \ldots, y_m)$ be a propositional logic formula in 3CNF. Let $\mathbf{S}$ be the schema that consists of all ternary relation names $C_{u_1u_2u_3}$ with $u_1u_2u_3 \in \{\mathsf{n}, \mathsf{p}\}^3$. Every clause in φ can be viewed as a fact over signature $\mathbf{S}$ by letting the u_i represent the polarities of the variables in the clause (n stands for *negative*, p stands for *positive*) and using the variables from φ as constants. For example, $\neg y_2 \vee y_1 \vee \neg y_3$ gives the fact $C_{\mathsf{npn}}(y_2, y_1, y_3)$. Thus, φ can be viewed as a database D_φ. What's more, we can build a database D that is independent of φ and such that $D_\varphi \to D$ if and only if φ is satisfiable. Using CSP parlance, we call D the *template for 3SAT*. It is actually easy to find D: use two constants 0 and 1 that represent truth values and add the fact $C_{u_1u_2u_3}(t_1, t_2, t_3)$ if the truth assignment t_1, t_2, t_3 to the three variables of a clause with polarities $u_1u_2u_3$ makes the clause true. For example, $C_{\mathsf{npn}}(t_1, t_2, t_3)$ is added for every $t_1t_2t_3 \in \{0, 1\}^3 \setminus \{101\}$.

How does this relate to the expressibility problem? Let q_s be D_φ viewed as a Boolean CQ and take the mappings $D_\varphi \to A(x)$ and $D \to A(x)$ where D_φ and D are viewed as CQs with x an arbitrary but fixed variable. Then $\mathbf{M}^-(\mathbf{M}(q_s))$ is (equivalent to) $D_\varphi \vee D$ with D_φ and D viewed as a Boolean CQs. By applying Theorem 5.8 where q_r is now simply $\mathbf{M}(q_s)$ since the ontology is empty, we obtain the following: q_s is CQ-expressible if and only if $q_s \to D_\varphi \vee D$, which is the case if and only if $q_s \to D_\varphi$ and $q_s \to D$, which in turn is the case if and only if $q_s \to D$, since $q_s \to D_\varphi$ always holds.

We now lift this simple reduction to $\forall\exists$-3SAT. Thus let

$$\varphi = \forall x_0 \cdots \forall x_n \exists y_0 \cdots \exists y_m \psi(x_0, \ldots, x_n, y_0, \ldots, y_m)$$

be a quantified Boolean formula with ψ in 3CNF. We construct an OBDA specification $(\emptyset, \mathbf{M}, \mathbf{S})$ with $\mathbf{M}$ a set of GAV mappings as well as a Boolean CQ q_s over schema $\mathbf{S}$ such that φ is true if and only if q_s is CQ-expressible in $(\emptyset, \mathbf{M}, \mathbf{S})$.

The universally quantified variables have a different status in the reduction as, unlike the existentially quantified variables, they are not represented by variables in q_s.

For example, the clause $(y_1 \vee \neg x_0 \vee \neg y_1)$ gives rise to the atom $C_{\mathsf{p}\neg x_0\mathsf{n}}(y_1, y_1)$. This is compensated by constructing the mappings in $\mathbf{M}$ so that $\mathbf{M}^{-}(\mathbf{M}(q_s))$ is now essentially a disjunction of (q_s and) exponentially many versions of the template D_φ, one for every truth assignment to the universally quantified variables. For example, such a template includes $C_{\mathsf{p}\neg x_0\mathsf{n}}(t_1, t_2)$ for *all* $t_1t_2 \in \{0,1\}^2$ if it represents a truth assignment that makes x_0 true and otherwise it includes $C_{\mathsf{p}\neg x_0\mathsf{n}}(t_1, t_2)$ for all $t_1t_2 \in \{0,1\}^2$ except 01. To achieve this, we use binary relations in the head of mappings instead of unary ones. In fact, we want $\mathbf{M}(q_s)$ to be of the form $\mathbf{M}(q_s) = \bigwedge_{i=0}^{n} r_i(y_0, y_1)$ and there will be two ways to translate each $r_i(y_0, y_1)$ backwards in the construction of $\mathbf{M}^{-}(\mathbf{M}(q_s))$, corresponding to the two possible truth values of the universally quantified variable x_i.

We now make the reduction precise. Let $U = \{x_0, \dots, x_n, \neg x_0, \dots, \neg x_n, \mathsf{n}, \mathsf{p}\}$. For every triple $u_1u_2u_3 \in U^3$ we include a relation $C_{u_1u_2u_3}$ in $\mathbf{S}$. The *arity* of $C_{u_1u_2u_3}$ is the number of positions in $u_1u_2u_3$ that are n or p. Additionally, $\mathbf{S}$ contains a binary relation Z which helps us to achieve that $\mathbf{M}(q_s)$ is of the intended form even when q_s admits non-trivial automorphisms.

We define q_s to encode φ. The existentially quantified variables of q_s are $y_0, \dots, y_m$. For every clause $\ell_1 \vee \ell_2 \vee \ell_3$ in ψ, we introduce an atom in q_s with the symbol $C_{u_1u_2u_3}$, where $u_i = \ell_i$ if ℓ_i contains a universally quantified variable, $u_i = \mathsf{p}$ if ℓ_i is a positive literal with an existentially quantified variable and $u_i = \mathsf{n}$ if ℓ_i is a negative literal with an existentially quantified variable. The variables that occur in this atom are the existentially qualified variables of the clause in the order of their appearance in the clause, see above for an example. Moreover, we add the atom $Z(y_0, y_1)$ to q_s, assuming w.l.o.g. that there are at least two existentially quantified variables in φ.

We now construct the GAV mappings in $\mathbf{M}$. For every universally quantified variable x_i of φ, we introduce three mappings with the same head $r_i(z_0, z_1)$:

1. In the first mapping $q_s'(z_0, z_1) \longrightarrow r_i(z_0, z_1)$, the body is q_s with y_0 and y_1 renamed to z_0 and z_1.

2. The body of the second mapping $\tau_i^0(z_0, z_1) \longrightarrow r_i(z_0, z_1)$ generates the part of the template that must be there when x_i is assigned truth value 0. The variables z_0 and z_1 represent the two elements of the template.

   The body τ_i^0 only contains the variables z_0 and z_1. For every $u_1u_2u_3 \in U^3$ and sequence $\mathbf{v} = v_1v_2\cdots$ over $\{z_0, z_1\}$ of the same length as the arity of $C_{u_1u_2u_3}$, we add the atom $C_{u_1u_2u_3}(\mathbf{v})$ to τ_i^0 if at least one of the following holds:

   a) $\neg x_i$ is among u_1, u_2 and u_3,

   b) $v_j = z_0$ and the j-th appearance of n or p in $u_1u_2u_3$ is n for some j,

   c) $v_j = z_1$ and the j-th appearance of n or p in $u_1u_2u_3$ is p for some j.

   We also add the atoms $Z(z_0, z_0)$, $Z(z_0, z_1)$, $Z(z_1, z_0)$ and $Z(z_1, z_1)$ to τ_i^0.

   (For example in τ_3^0 we add the atoms $C_{\mathsf{p}x_3\mathsf{n}}(z_0, z_0)$, $C_{\mathsf{p}x_3\mathsf{n}}(z_1, z_0)$, and $C_{\mathsf{p}x_3\mathsf{n}}(z_1, z_1)$, but not $C_{\mathsf{p}x_3\mathsf{n}}(z_0, z_1)$. We add all $C_{\mathsf{p}\neg x_3\mathsf{n}}(z_0, z_0)$, $C_{\mathsf{p}\neg x_3\mathsf{n}}(z_0, z_1)$, $C_{\mathsf{p}\neg x_3\mathsf{n}}(z_1, z_0)$ and $C_{\mathsf{p}\neg x_3\mathsf{n}}(z_1, z_1)$. Also, we add $C_{x_2x_3\mathsf{p}}(z_1)$ but not $C_{x_2x_3\mathsf{p}}(z_0)$.)

3. The body $\tau_i^1(z_0, z_1)$ of the third mapping $\tau_i^1(z_0, z_1) \longrightarrow r_i(z_0, z_1)$ is dual to $\tau_i^0(z_0, z_1)$ in that it encodes the case where x_i is true. This means the definition is as above with the difference that in Condition 2a, the literal x_i, and not $\neg x_i$, is required to be among u_1, u_2 and u_3.

Lemma 5.16. *φ is true if and only if q_s is CQ-expressible in $(\emptyset, \mathbf{M}, \mathbf{S})$.*

Proof. By Theorem 5.8, it suffices to show that φ is true if and only if $\mathbf{M}^-(\mathbf{M}(q_s)) \subseteq q_s$, that is, if and only if there is a homomorphism from q_s to every disjunct of $\mathbf{M}^-(\mathbf{M}(q_s))$.

We first describe the UCQ $\mathbf{M}^-(\mathbf{M}(q_s))$. First observe that $\mathbf{M}(q_s)$ is indeed $\bigwedge_{i=0}^n r_i(y_0, y_1)$: The mapping $q_s' \longrightarrow r_i(z_0, z_1)$ has a match at (y_0, y_1) for every i and it has no other matches since $Z(z_0, z_1)$ is in q_s' and $Z(y_0, y_1)$ is the only atom in q_s the contains Z. The mappings $\tau_i^k(z_0, z_1) \longrightarrow r_i(z_0, z_1)$ do not match anywhere in q_s for $k \in \{0, 1\}$, since the atoms $Z(z_0, z_0)$, $Z(z_0, z_1)$, $Z(z_1, z_0)$ and $Z(z_1, z_1)$ all appear in the body of these mappings. There are 3^{n+1} disjuncts in $\mathbf{M}^-(\mathbf{M}(q_s))$, one for every combination of $n + 1$ choices of the three different mappings with head $r_i(z_0, z_1)$, for every $i \in \{0, \dots, n\}$. We now prove the lemma.

"$\Rightarrow$". Assume that φ is true. We want to show that there is a homomorphism from q_s into every CQ in $\mathbf{M}^-(\mathbf{M}(q_s))$. Pick an arbitrary CQ p in $\mathbf{M}^-(\mathbf{M}(q_s))$. Such a disjunct corresponds of a choice of one of the bodies q_s', τ_i^0, or τ_i^1 for each $i \in \{0, \dots, n\}$.

First consider the case where for some i we choose q_s'. In that case p contains an isomorphic copy of q_s and thus we are done.

Now consider the case where for no i we choose a mapping with body q_s', that is, for every i we choose τ_i^0 or τ_i^1. This corresponds to an assignment t of the truth values 0 and 1 to the variables $x_0, \dots, x_n$. Because $\varphi = \forall x_0, \dots, x_n \exists y_0, \dots, y_m \psi$ is true we can extend t to an assignment for $x_0, \dots, x_n, y_0, \dots, y_m$ that makes ψ true. We define the homomorphism h from q_s to p such that $h(y_j) = y_{t(y_j)}$ for all $j \in \{0, \dots, m\}$. We need to verify that h is a homomorphism. All atoms with the symbol Z are preserved as by definition of the τ_i^0 and τ_i^1, there is a Z atom in p for any pair over $\{y_0, y_1\}$. Consider then any atom $C_{u_1 u_2 u_3}(\mathrm{y})$ from q_s. There is a corresponding clause $\ell_1 \vee \ell_2 \vee \ell_3$ in ψ that is true under the assignment t. It follows that one of the literals ℓ_1, ℓ_2, ℓ_3 is true under t. Let ℓ_k be this literal. We make a case distinction:

If $\ell_k = x_i$ is a universally quantified variable, then $t(x_i) = 1$ and hence p contains τ_i^1. By (the implicit) Condition 3a, from the definition of $\mathbf{M}$, it follows that τ_i^1 contains the atom $C_{u_1 u_2 u_3}(y_{t(\mathrm{y})})$.

In the case where $\ell_k = \neg x_i$, we use the same argument and Condition 2a.

If $\ell_k = y_j$ is an existentially quantified variable, then $u_k = \mathrm{p}$ and $t(y_j) = 1$. Hence, $h(y_j) = y_1$ and from Condition (2c) or (3c), the atom $C_{u_1 u_2 u_3}$ is in τ_0^0 and in τ_0^1, thus in p (since we can assume w.l.o.g. that there is at least one universally quantified variable).

The case where $\ell_k = \neg y_j$ is analogous, using Conditions (2b) or (3b).

"$\Leftarrow$". Assume that there is a homomorphism from q_s into every disjunct of $\mathbf{M}^-(\mathbf{M}(q_s))$. We want to show that φ is true. So take any assignment t for $x_0, \dots, x_n$. We need to extend t to an assignment t' for $x_0, \dots, x_n, y_0, \dots, y_m$ that makes ψ true. Consider the disjunct p_t of $\mathbf{M}^-(\mathbf{M}(q_s))$ that arises from choosing the mapping $\tau_i^{t(x_i)}(z_0, z_1) \longrightarrow r_i(z_0, z_1)$

for each $i = 0, \ldots, n$. By assumption, there is a homomorphism h from q_s to p_t. Clearly, p_t contains only the variables y_0 and y_1. We define t' such that $t'(x_i) = t(x_i)$, $t'(y_j) = 0$ if $h(y_j) = y_0$, and $t'(y_j) = 1$ if $h(y_j) = y_1$.

It remains to be shown that ψ is true under t'. Take an arbitrary clause $\ell_1 \vee \ell_2 \vee \ell_3$ in ψ. Consider the corresponding atom $C_{u_1u_2u_3}(\mathbf{y})$ in q_s. As h is a homomorphism, $C_{u_1u_2u_3}(h(\mathbf{y}))$ is in p_t. By definition of p_t this means that $C_{u_1u_2u_3}(h(\mathbf{y}))$ is contained in $\tau_i^{t(x_i)}$ for some i. Hence one of the conditions (a), (b) or (c) from Point 2 or 3 of the construction of $\mathbf{M}$ is satisfied.

If (a) is satisfied and $t(x_i) = 0$, then $\neg x_i$ is among u_1, u_2, u_3. It follows that then $\neg x_i$ is a literal in $\ell_1 \vee \ell_2 \vee \ell_3$ and hence the clause is true under t' because $t'(x_i) = t(x_i)$. In case (a) is satisfied and $t(x_i) = 1$, we can reason analogously.

If (b) is satisfied, then $h(y_i) = y_0$ for some y_i in $\overline{y}$, and the i-th appearance of either n or p in $u_1u_2u_3$ is n. Hence $\neg y_i$ is a literal in $\ell_1 \vee \ell_2 \vee \ell_3$ which makes the clause true because $t'(y_1) = 0$ since $h(y_i) = y_0$. We reason analogously in the case where (c) is satisfied. □

5.4 Expressibility in $\mathcal{ELHI}$: Upper Bound for Rooted Queries

We show that the expressibility problem in $[\mathcal{ELHI}, \text{GAV}]$ is in coNExpTime when the source query is a rooted UCQ.

Theorem 5.17. *The rUCQ-to-UCQ expressibility problem for* $[\mathcal{ELHI}, \text{GAV}]$ *is solvable in* coNExpTime.

To prepare for lifting the result from expressibility to verification later, we actually establish a slightly more general result as needed. Note that the following implies Theorem 5.17 since, by Theorem 5.8, we can simply use $\mathbf{M}(q_s)$ for q_t.

Theorem 5.18. *Given an OBDA setting $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ from $[\mathcal{ELHI}, GAV]$, an rUCQ q_s over* $\mathbf{S}$, *and a UCQ q_t over* $\mathsf{sch}(\mathbf{M})$, *it is in* coNExpTime *to decide whether* $\mathbf{M}^-(q_r) \subseteq_{\mathbf{S}} q_s$, *where q_r is an infinitary UCQ-rewriting of the OMQ $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$.*

To prove Theorem 5.18, we now describe a NExpTime algorithm for deciding the complement of the problem described there: we want to check whether $\mathbf{M}^-(q_r) \not\subseteq q_s$, that is, whether there is a CQ p in the UCQ $\mathbf{M}^-(q_r)$ such that $q \not\rightarrow p$ for all CQs q in q_s. Because all rewritings of Q are equivalent, it suffices to prove the theorem for any particular infinitary UCQ-rewriting q_r of Q. We choose to work with the canonical one introduced at the beginning of the characterizations section. The algorithm is as follows:

1. Guess a $\mathsf{sch}(\mathbf{M})$-ABox $\mathcal{A}$ such that $|\mathsf{ind}(\mathcal{A})| \leq |q_t| + |q_t| \cdot |\mathcal{T}|^{|q_s|+1}$ and a tuple $\mathbf{a}$ in $\mathcal{A}$ of the same arity as q_t.

2. Verify that $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ to make sure that $(\mathcal{A}, \mathbf{a})$ viewed as a CQ is in the UCQ q_r. This can be done by an algorithm that is exponential in $|\mathcal{T}|$ and $|q_t|$, but only polynomial in $|\mathcal{A}|$; see for example [KL07]. Hence, the overall running time is single exponential in the size of the original input.

3. Guess a disjunct p from the UCQ $\mathbf{M}^-(\mathcal{A}, \mathbf{a})$ by guessing, for each fact α in $\mathcal{A}$, a suitable mapping from $\mathbf{M}$. Note that both $\mathcal{A}$ and p are of single exponential size.

4. Verify that $q \not\twoheadrightarrow p$ for all CQs q in the rUCQ q_s. This can be done in single exponential time using brute force.

This is clearly a NExpTime algorithm.

Lemma 5.19. *The algorithm decides the complement of the problem in Theorem 5.18.*

Proof. It is easy to verify the soundness part: a successful run of the algorithm identifies p as a CQ in $\mathbf{M}^-(q_r)$ such that for any disjunct q of q_s, $q \not\twoheadrightarrow p$, which yields $\mathbf{M}^-(q_r) \not\subseteq_\mathsf{S} q_s$.

For the completeness direction assume that $\mathbf{M}^-(q_r) \not\subseteq_\mathsf{S} q_s$. This means there is an ABox $\mathcal{A}$ and tuple $\mathbf{a}$ such that $(\mathcal{A}, \mathbf{a})$ is in the canonical rewriting q_r of Q and there is p in $\mathbf{M}^-(\mathcal{A}, \mathbf{a})$ such that for all q in q_s we have $q \not\to p$. We show how to obtain an ABox $\mathcal{A}''$ that the algorithm can choose in step 1 in order to accept.

By Lemma 2.3, there exists an pseudo tree-shaped ABox $\mathcal{A}'$ of core-size $|q_t|$ and branching degree at most $|\mathcal{T}|$ and tuple $\mathbf{a}'$ in the core of $\mathcal{A}'$ such that $\mathbf{a}' \in \mathsf{cert}_Q(\mathcal{A}')$ and there is a homomorphism $(\mathcal{A}', \mathbf{a}') \to (\mathcal{A}, \mathbf{a})$.

Define $\mathcal{A}''$ to be obtained from the pseudo tree-shaped ABox $\mathcal{A}'$ by removing all assertions that contain at least one individual that has a distance larger than $|q_s|$ from the core. Furthermore, we add all assertions $A(b)$ and $r(b, b)$ for all A and r in $\mathsf{sch}(\mathbf{M})$ and for all individuals b that have exactly distance $|q_s|$ from the core.

The size of the resulting ABox $\mathcal{A}''$ is at most $|q_t| + |q_t| \cdot |\mathcal{T}|^{|q_s|}$, so $\mathcal{A}''$ can be chosen in Step 1.

Claim: Step 2 succeeds, that is, $\mathbf{a}' \in \mathsf{cert}_Q(\mathcal{A}'')$.

Proof of claim: Define a homomorphism $h : \mathcal{A}' \to \mathcal{A}''$ that is the identity on individuals of distance at most $|q_s|$ from the core and maps an individual b of distance more than $|q_s|$ from the core to the unique b' of distance precisely $|q_s|$ such that b is in the subtree rooted at b'. This indeed defines a homomorphism because in $\mathcal{A}''$ all symbols in $\mathsf{sch}(\mathbf{M})$ are true at each individual with distance $|q_s|$. Now $\mathbf{a}' \in \mathsf{cert}_Q(\mathcal{A}'')$ follows from $\mathbf{a}' \in \mathsf{cert}_Q(\mathcal{A}')$ because h is a homomorphism that fixes the tuple $\mathbf{a}'$.

We describe the query p' in $\mathbf{M}^-(\mathcal{A}'')$ that can be chosen in step 3. For every fact in $\mathcal{A}''$ that is also in $\mathcal{A}'$ choose the same mapping that was chosen to obtain p from $\mathcal{A}$. For all other facts we choose an arbitrary suitable mapping.

Claim: Step 4 succeeds, that is, $q \not\to p'$ for all q in q_s.

Proof of claim: Assume towards a contradiction that there is a homomorphism $g' : q \to p'$ for some q in q_s. Because $\mathcal{A}''$ and $\mathcal{A}'$ are equal when restricted to variables of distance less than $|q_s|$ from $\mathbf{a}'$ it follows that p and p' are equal when restricted to variables of distance less than $|q_s|$ from $\mathbf{a}'$. This is due to the fact that the distance of two variables from $\mathcal{A}$ cannot decrease in p in $\mathbf{M}^-(\mathcal{A})$, and similarly for $\mathcal{A}''$ and p'.

Because $|q| \leq |q_s|$ and q is rooted it follows that all constants in the image of $g' : q \to p'$ have distance less than $|q_s|$ from $\mathbf{a}$ in p. Since $\mathbf{a}$ lies in the core, these constants have also distance less than $|q_s|$ from the core. Because p and p' are equal when restricted to constants of distance less than $|q_s|$ from the core, there is a homomorphism $g : q \to p$, which contradicts our assumption on p. $\square$

5.5 Expressibility in $\mathcal{ELHI}$: Upper Bound for Unrestricted Queries

We consider the UCQ-to-UCQ expressibility problem in $[\mathcal{ELHI}, \text{GAV}]$, that is, we drop the assumption from the previous section that the source query is rooted. This increases the complexity from coNExpTime to 2-ExpTime. Note that similar effects have been observed in the context of different reasoning problems in [Lut08; Bie+16]. In this section, we show the upper bound.

Theorem 5.20. *In $[\mathcal{ELHI}, \text{GAV}]$, the UCQ-to-UCQ expressibility problem is in* 2ExpTime.

As in the rooted case, we again prove a slightly more general result that can be reused when studying the verification problem. We can obtain Theorem 5.20 from the following by setting $q_t = \mathbf{M}(q_s)$ and applying Theorem 5.8.

Theorem 5.21. *Given an OBDA setting $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ from $[\mathcal{ELHI}, \text{GAV}]$ a UCQ q_s over $\mathbf{S}$, and a UCQ q_t over $\mathsf{sch}(\mathbf{M})$, it is in 2ExpTime to decide whether $\mathbf{M}^-(q_r) \subseteq_{\mathbf{S}} q_s$, where q_r is an infinitary UCQ-rewriting of the OMQ $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$.*

To prove Theorem 5.21, we start by choosing a suitable UCQ-rewriting q_r. Instead of working with the canonical infinitary UCQ-rewriting, here we prefer to use the UCQ that consists of all pairs $(\mathcal{A}, \mathbf{a})$ viewed as a CQ and where $\mathcal{A}$ is a pseudo tree-shaped $\mathsf{sch}(\mathbf{M})$-ABox that satisfies $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ and is of the dimensions stated in Lemma 2.3, that is, the core of $\mathcal{A}$ is not larger than $|q|$ and the outdegree of $\mathcal{A}$ is not larger than $|\mathcal{T}|$. Due to that lemma, q_r clearly is an infinitary UCQ-rewriting of Q.

We give a decision procedure for the complement of the problem in Theorem 5.21. We thus have to decide whether there is a pseudo-tree shaped $\mathsf{sch}(\mathbf{M})$-ABox $\mathcal{A}$ of the mentioned dimensions, an $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$, and a CQ p in the UCQ $\mathbf{M}^-(\mathcal{A}, a)$ such that $q \not\rightarrow p$ for all CQs q in q_s. This can be done by constructing a TWAPA $\mathfrak{A}$ on finite trees (introduced in Section 2.9) that accepts precisely those trees that represent a triple $(\mathcal{A}, \mathbf{a}, p)$ with the components as described above, and then testing whether the language accepted by $\mathfrak{A}$ is empty.

In the following, we detail this construction. We reuse some encodings and notation from a TWAPA construction that is employed in [Bie+16] to decide OMQ containment as this saves us from redoing certain routine work. We encode triples $(\mathcal{A}, \mathbf{a}, p)$ as finite $(|\mathcal{T}| \cdot |q_t|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees, where Σ_ε is the alphabet used for labeling the root node and Σ_N is for non-root nodes. These alphabets are different because the root of a tree represents the core part of a pseudo tree-shaped ABox whereas each non-root node represents a single constant of the ABox that is outside the core. Let $\mathsf{C}_{\mathsf{core}}$ be a fixed set of $|q_t|$ constants. Formally, the alphabet Σ_ε is the set of all triples $(\mathcal{B}, \mathbf{a}, \mu)$ where $\mathcal{B}$ is a

$\mathsf{sch}(\mathbf{M})$-ABox of size at most $|q_t|$ that uses only constants from $\mathsf{C}_{\mathsf{core}}$, $\mathbf{a}$ is a tuple over $\mathsf{C}_{\mathsf{core}}$ whose length matches the arity of q_s, and μ associates every fact α in $\mathcal{B}$ with a mapping $\mu(\alpha) \in \mathbf{M}$ that is suitable for α. The alphabet Σ_N consists of all triples (Θ, M, μ) where $\Theta \subseteq (\mathsf{N}_\mathsf{C} \cap \mathsf{sch}(\mathbf{M})) \uplus \{r, r^- \mid r \in \mathsf{N}_\mathsf{R} \cap \mathsf{sch}(\mathbf{M})\} \uplus \mathsf{C}_{\mathsf{core}}$ contains exactly one (potentially inverse) role and at most one element of $\mathsf{C}_{\mathsf{core}}$, $M \in \mathbf{M}$ is a mapping suitable for the fact $r(a, b)$ with r the unique role name in Θ, and μ assigns to each $A \in \Theta$ a mapping $\mu(A) \in \mathbf{M}$ suitable for the fact $A(a)$.[3] In the following, a *labeled tree* generally means a $(|\mathcal{T}| \cdot |q_t|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled tree.

A labeled tree is *proper* if

(i) the root node is labeled with a symbol from Σ_ε,

(ii) each child of the root is labeled with a symbol from Σ_N that contains an element of $\mathsf{C}_{\mathsf{core}}$

(iii) every other non-root node is labeled with a symbol from Σ_N that contains no constant name

(iv) every non-root node has at most $|\mathcal{T}|$ successors and

(v) for every $a \in \mathsf{C}_{\mathsf{core}}$, the root node has at most $|\mathcal{T}|$ successors whose label includes a.

A proper labeled tree (T, L) with $L(\varepsilon) = (\mathcal{B}, \mathbf{a}, \mu)$ *encodes* the triple $(\mathcal{A}, \mathbf{a}, p)$ where

$$\begin{aligned}\mathcal{A} \;=\; & \mathcal{B} \cup \{A(x) \mid A \in \Theta(x)\} \\ & \cup \{r(b, x) \mid \{b, r\} \subseteq \Theta(x)\} \cup \{r(x, b) \mid \{b, r^-\} \subseteq \Theta(x)\} \\ & \cup \{r(x, y) \mid r \in \Theta(y), y \text{ is a child of } x, \Theta(x) \in \Sigma_N\} \\ & \cup \{r(y, x) \mid r^- \in \Theta(y), y \text{ is a child of } x, \Theta(x) \in \Sigma_N\},\end{aligned}$$

$\Theta(x)$ denoting Θ when $L(x) = (\Theta, M, \mu)$ (and undefined otherwise), and where p is the CQ from $\mathbf{M}^-(\mathcal{A})$ that can be obtained by choosing for every fact in $\mathcal{A}$ the suitable mapping from $\mathbf{M}$ assigned to it by L.

The desired TWAPA $\mathfrak{A}$ is obtained as the intersection of two TWAPA $\mathfrak{A}_1$ and $\overline{\mathfrak{A}}_2$, where $\mathfrak{A}_1$ accepts exactly the proper labeled trees (T, L) that encode a pair $(\mathcal{A}, \mathbf{a}, p)$ with $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ and $\overline{\mathfrak{A}}_2$ is obtained as the complement of an automaton $\mathfrak{A}_2$ that accepts a proper labeled tree (T, L) encoding a pair $(\mathcal{A}, \mathbf{a}, p)$ if and only if $q \to p$ for some CQ q in q_s. In fact, the automaton $\mathfrak{A}_1$ is what we can reuse from [Bie+16], see Point 1 in Proposition 13 there. The only difference is that our trees are decorated in a richer way, so in our case the TWAPA ignores the part of the labeling that is concerned with mappings from $\mathbf{M}$. The number of states of $\mathfrak{A}_1$ is single exponential in $|q_t|$ and $|\mathcal{T}|$.

We sketch the construction of the automaton $\mathfrak{A}_2$ for a single CQ q of q_s (the general case can be dealt with using union). Let $q_1, \ldots, q_k$ be the maximal connected components of q. We define automata $\mathfrak{A}_{2,1}, \ldots, \mathfrak{A}_{2,k}$ where $\mathfrak{A}_{2,i}$ accepts (T, L) encoding $(\mathcal{A}, \mathbf{a}, p)$ if and only if $q_i \to p$, and then intersect to obtain $\mathfrak{A}_2$. Let (T, L) be a proper labeled tree. A

[3]Here, a and b are arbitrary but fixed constants.

set $T' \subseteq T$ is a *subtree* of T if for any $s, t \in T'$, all nodes from T that are on the shortest (undirected) path from s to t are in T'. We use (T', L) to denote the restriction of (T, L) to T' and $\mathbf{M}^{-}(T', L)$ to denote the subquery of p which contains only the atoms in p that can be derived from the part of $\mathcal{A}$ generated by the subtree (T', L) of (T, L).

To define $\mathfrak{A}_{2,i}$, let C denote the set of all labeled trees (T', L) of size at most $|q_i|$ such that there is a proper labeled tree (T, L) and $q_i \longrightarrow \mathbf{M}^{-}(T', L)$ with a homomorphism that only needs to respect the answer variables from q that actually occur in q_i (if there are any, then T' must thus contain the root of T). The automaton $\mathfrak{A}_{2,i}$ is then constructed such that it accepts a proper labeled tree (T, L) if and only if it contains a subtree from C. It should be clear that such an automaton can be constructed using only single exponentially many states. Moreover, it can be verified that $\mathfrak{A}_{2,i}$ accepts exactly the desired trees. We obtain an overall automaton with single exponentially many states which together with the ExpTime-complete emptiness problem of TWAPA gives Theorem 5.21.

5.6 Verification in $\mathcal{ELHI}$: Upper Bounds

We show that in $[\mathcal{ELHI}, \text{GAV}]$, the complexity of the verification problem is not higher than the complexity of the expressibility problem both in the rooted and in the unrestricted case.

Theorem 5.22. *In $[\mathcal{ELHI}, \text{GAV}]$,*

1. *the rUCQ-to-UCQ verification problem can be decided in coNExpTime.*
2. *the UCQ-to-UCQ verification problem can be decided in 2ExpTime.*

Recall the characterization of realizations from Theorem 5.6: a UCQ q_t is a realization of q_s if and only if $q_s \equiv \mathbf{M}^{-}(q_r)$, where q_r is a rewriting of the OMQ $(\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$. The inclusion $q_s \supseteq \mathbf{M}^{-}(q_r)$ is already treated by Theorems 5.18 and 5.21 and thus it remains to show that the converse inclusion can be decided in the relevant complexity class. We show that this is actually possible in ExpTime, even in the unrestricted case. We thus aim to prove the following.

Theorem 5.23. *Given an OBDA setting $\mathcal{S} = (\mathcal{T}, \mathbf{M}, \mathbf{S})$ from $[\mathcal{ELHI}, \text{GAV}]$, a UCQ q_s over $\mathbf{S}$, and a UCQ q_t over $\mathsf{sch}(\mathbf{M})$, it is in ExpTime to decide whether $q_s \subseteq_{\mathbf{S}} \mathbf{M}^{-}(q_r)$, where q_r is an infinitary UCQ-rewriting of the OMQ $Q = (\mathcal{T}, \mathsf{sch}(\mathbf{M}), q_t)$.*

For what follows, it is convenient to assume that the TBox $\mathcal{T}$ is in normal form, see Section 2.6. It is easy to verify that for the verification problem, we can w.l.o.g. assume the involved ontology to be in normal form.

We again start by choosing a suitable concrete infinitary UCQ-rewriting to use for q_r. As in the previous section, we would like to use CQs derived from pseudo tree-shaped ABoxes of certain dimension that entail an answer to the OMQ Q, as sanctioned by Lemma 2.3. This time, we use the stronger version of said lemma, so we use for q_r the set of all pairs $(\mathcal{A}, \mathbf{a})$ viewed as a CQ where $\mathcal{A}$ is a pseudo tree-shaped sch(**M**)-ABox with core C that satisfies

$$\mathbf{a} \in \mathsf{cert}_Q(C \cup \{A(a) \mid \mathcal{A}, \mathcal{T} \models A(a),\ a \in \mathsf{ind}(C)\}) \tag{5.2}$$

and is of the dimensions stated in Lemma 2.3.

For deciding $q_s \subseteq_\mathsf{S} \mathbf{M}^-(q_r)$, we need to show that for every disjunct q in q_s there is a disjunct p in $\mathbf{M}^-(q_r)$ such that $p \rightarrow q$. We can do this for every disjunct q of q_s separately. Hence let q be such a disjunct. To find a CQ p in $\mathbf{M}^-(q_r)$ with $p \rightarrow q$, we again aim to utilize TWAPA. As in the previous section, let C_core be a fixed set of $|q_t|$ individuals. A *homomorphism pattern* for q_t is a function λ that maps every variable y in q_t to a pair $(a, o) \in \mathsf{C}_\mathsf{core} \times \{\mathsf{core}, \mathsf{subtree}\}$. Informally, λ is an abstract description of a homomorphism h from q_t to the universal model of a pseudo-tree ABox $\mathcal{A}$ and $\mathcal{T}$ (assume that the core of $\mathcal{A}$ uses only constants from C_core) such that $h(x) = a$ when $\lambda(x) = (a, \mathsf{core})$ and $h(x)$ is an element in the anonymous subtree below a when $\lambda(x) = (a, \mathsf{subtree})$.

We build one TWAPA $\mathfrak{A}^\lambda$ for every homomorphism pattern λ (there are single exponentially many). These TWAPA again run on $(|\mathcal{T}| \cdot |q_t|)$-ary $\Sigma_\varepsilon \cup \Sigma_N$-labeled trees that encode a triple $(\mathcal{A}, \mathbf{a}, p)$, defined exactly as in the previous section and from now on are only referred to as labeled trees. We also use the same notion of properness as in the previous section. The TWAPA $\mathfrak{A}^\lambda$ shall accept exactly those labeled trees (T, L) that are proper and encode a triple $(\mathcal{A}, \mathbf{a}, p)$ such that

1. there is a homomorphism h from $q_t(\mathbf{x})$ to the universal model of $\mathcal{A}$ and $\mathcal{T}$ that satisfies $h(\mathbf{x}) = \mathbf{a}$ and follows the homomorphism pattern λ, which means that $\lambda(y) = (h(y), \mathsf{core})$ if $h(y) \in \mathsf{C}_\mathsf{core}$ and $\lambda(y) = (c, \mathsf{subtree})$, if $h(y)$ lies in an anonymous subtree of the universal model of $\mathcal{A}$ and $\mathcal{T}$ whose root is $c \in \mathsf{C}_\mathsf{core}$, and

2. $p \rightarrow q$.

Note that it would be sufficient to demand in Point 1 that $\mathbf{a} \in \mathsf{cert}_Q(\mathcal{A})$ and recall that, in the previous section, we have reused an automaton from [Bie+16] which checks exactly this condition. That automaton, however, has exponentially many states because it is built using a construction known under various names such as query splitting, forest decomposition, and squid decomposition. To attain an ExpTime upper bound, though, the automaton $\mathfrak{A}^\lambda$ can only have polynomially many states. This is in fact the reason why we use the stronger version of Lemma 2.3. It is easy to see that Condition 1 here holds for some λ if and only if Condition (5.2) holds.

Hence Condition 1 guarantees that $(\mathcal{A}, \mathbf{a})$ is in the rewriting q_r. Together with the second point and the conditions on the encoded tuples $(\mathcal{A}, \mathbf{a}, p)$ this then means that there is a λ such that $\mathfrak{A}^\lambda$ accepts a tree encoding $(\mathcal{A}, \mathbf{a}, p)$ if and only if p is a disjunct in $\mathbf{M}^-(q_r)$ such that $p \rightarrow q$.

We construct the automaton $\mathfrak{A}^\lambda$ as the intersection of three automata $\mathfrak{A}_\mathsf{proper}$, $\mathfrak{A}_1^\lambda$, and $\mathfrak{A}_2$, where $\mathfrak{A}_\mathsf{proper}$ accepts if the input tree is proper, $\mathfrak{A}_1^\lambda$ accepts trees that encode a triple $(\mathcal{A}, \mathbf{a}, p)$ that satisfy Condition 1 for λ and $\mathfrak{A}_2$ accepts trees that encode a triple $(\mathcal{A}, \mathbf{a}, p)$ that satisfy Condition 2. We obtain single exponentially many automata $\mathfrak{A}^\lambda$, one for every homomorphism pattern λ, with polynomially many states each and we answer 'yes' if any of the automata recognizes a non-empty language. By Theorem 2.7, this gives an ExpTime algorithm, finishing the proof of Theorem 5.23. It remains to construct the automata and prove their correctness. The construction of $\mathfrak{A}_{\text{proper}}$ is easy, we leave out the details.

Definition of $\mathfrak{A}_1^\lambda$. To define $\mathfrak{A}_1^\lambda$, we first introduce some notation. Let ROL be the set of roles that appear in $\mathcal{T}$ or $\mathsf{sch}(\mathbf{M})$. Let CN the set of concept names that appear in $\mathcal{T}$ or $\mathsf{sch}(\mathbf{M})$ and let $\mathsf{tp} = 2^{\mathsf{CN}}$. Let U be the set of all partial functions from the variables in q_t to the set $\{\mathsf{core}, \mathsf{subtree}\}$. We define a relation $R \subseteq \mathsf{tp} \times U$ such that $(t, f) \in R$ if and only if there is a homomorphism g from q_t restricted to variables in the domain of f to the universal model of $\mathcal{T}$ and $\{A(a) \mid A \in t\}$ such that $g(y) = a$ if and only if $f(y) = \mathsf{core}$. The relation R can be computed in exponential time, since for every pair $(t, f) \in \mathsf{tp} \times U$, one can construct a simple TWAPA with polynomially many states, that checks whether $(t, f) \in R$. We do not detail the construction, but give a rough sketch instead: The input tree encodes a tree-shaped ABox. The TWAPA guesses a homomorphism from q_t to the input tree in a top-down fashion that respects f, that is, a homomorphism that maps every $x \in \mathsf{var}(q_t)$ with $f(x) = \mathsf{core}$ to the root of the encoded input tree and every other variable to a non-root individual. It then checks whether all assertions in the range of the homomorphism can be traced back to assertions of the form $A(a)$, where $A \in t$ and a is the root of the input tree.

Let $\mathfrak{A}_1^\lambda = (S, \delta, \Sigma_\varepsilon \uplus \Sigma_N, s_0, c)$ where

$$\begin{aligned} S = \{s_0\} &\uplus \{s_b^A \mid A \in \mathsf{CN} \text{ and } b \in \mathsf{C}_{\mathsf{core}}\} \\ &\uplus \{s_{r,b}^A \mid A \in \mathsf{CN} \text{ and } r \in \mathsf{ROL} \text{ and } b \in \mathsf{C}_{\mathsf{core}}\} \\ &\uplus \{s_r^A \mid A \in \mathsf{CN} \text{ and } r \in \mathsf{ROL}\} \\ &\uplus \{s^A \mid A \in \mathsf{CN}\} \end{aligned}$$

and $c(s) = 1$ for every $s \in S$, i.e. precisely the finite runs are accepting. All states besides s_0 are used to check whether a certain fact $A(a)$ is entailed in the universal model of $\mathcal{A}$ and $\mathcal{T}$. Following Lemma 2.4, this can be done by checking the existence of an appropriate derivation tree. The state s_b^A checks whether $\mathcal{A}, \mathcal{T} \models A(b)$. The state $s_{r,b}^A$ checks whether we are in an r-child d of b such that $\mathcal{A}, \mathcal{T} \models A(d)$. The state s_r^A checks whether the current node d is an r-child such that $\mathcal{A}, \mathcal{T} \models A(d)$. The state s^A checks whether for the current node d we have $\mathcal{A}, \mathcal{T} \models A(d)$.

We now describe the transition function δ. To define $\delta(s_0, l)$, where $l = (\mathcal{B}, \mathbf{a}, \mu) \in \Sigma_\varepsilon$ we distinguish cases depending on whether $\mathcal{B}$ and $\mathbf{a}$ are compatible with the homomorphism pattern λ. That is we check whether the following two conditions are fulfilled:

- $\lambda(x_i) = (a_i, \mathsf{core})$ for all $i \in \{1, \ldots, \mathsf{ar}(q_t)\}$.
- For every $r(z_1, z_2)$ in q_t such that $\lambda(z_i) = (b_i, \mathsf{core})$ we have that $r(b_1, b_2) \in \mathcal{B}$.

If these conditions are not fulfilled then we set $\delta(s_0, l) = \mathsf{false}$, meaning that the automaton $\mathfrak{A}_1^\lambda$ immediately rejects the input tree. If these conditions are fulfilled then define $\delta(s_0, l)$

to be:

$$\bigwedge_{\substack{z\in\mathsf{var}(q_t)\\ \lambda(z)=(b,\mathsf{core})}} \bigwedge_{A(z)\in q_t} \langle 0\rangle s_b^A \quad \wedge \tag{5.3}$$

$$\bigwedge_{\substack{Z\subseteq\mathsf{var}(q_t)\\ Z=\lambda^{-1}(\{b\}\times\{\mathsf{core},\mathsf{subtree}\})\\ Z\neq\emptyset}} \bigvee_{\substack{t\in\mathsf{tp}\\ (t,f)\in R\\ f=\pi_2\circ\lambda|_Z}} \bigwedge_{A\in t} \langle 0\rangle s_b^A \tag{5.4}$$

Here the $\lambda|_Z$ denotes the restriction of λ to the variables in Z and π_2 denotes the projection to the second component. The conjunction in the first line makes sure that the concept names needed for variables of q_t that are mapped to $\mathsf{C}_{\mathsf{core}}$ are derived. The second line assures that for all variables of q_t that are mapped to a fresh subtree generated in the universal model of $\mathcal{A}$ and $\mathcal{T}$, there is actually a type t derived at the root $b\in\mathsf{C}_{\mathsf{core}}$ that generates a suitable tree.

The following transitions are then used to check for derivations of concept names.

For $l\in\Sigma_\varepsilon$, let:

$$\begin{aligned}\delta(s_b^A,l) = & \bigvee_{\mathcal{T}\models B_1\sqcap\cdots\sqcap B_n\sqsubseteq A} \langle 0\rangle q_b^{B_1}\wedge\ldots\wedge\langle 0\rangle s_b^{B_n} \ \vee\\ & \bigvee_{\substack{\exists s.B\sqsubseteq A\in\mathcal{T}\\ \mathcal{T}\models r\sqsubseteq s}} \bigvee_{\substack{b'\in\mathsf{C}_{\mathsf{core}}\\ r(b,b')\in\mathcal{B}}} \langle 0\rangle s_{b'}^B \ \vee\\ & \bigvee_{\substack{\exists s.B\sqsubseteq A\in\mathcal{T}\\ \mathcal{T}\models r\sqsubseteq s}} \bigvee_{1\leq i\leq|\mathcal{T}|\cdot|q_t|} \langle i\rangle s_{r,b}^B\end{aligned}$$

For $l\in\Sigma_N$ and $\{r,b\}\subseteq l$ let:

$$\delta(s_{r,b}^A,l)=\langle 0\rangle s^A.$$

For $l\in\Sigma_N$ and $r\in l$ let

$$\delta(s_r^A,l)=\langle 0\rangle s^A.$$

For $l\in\Sigma_N$ with role $r\in l$ such that l contains no constant of $\mathsf{C}_{\mathsf{core}}$, let:

$$\begin{aligned}\delta(s^A,l) = & \bigvee_{\mathcal{T}\models B_1\sqcap\cdots\sqcap B_n\sqsubseteq A} \langle 0\rangle s^{B_1}\wedge\ldots\wedge\langle 0\rangle s^{B_n} \ \vee\\ & \bigvee_{\substack{\exists s^-.B\sqsubseteq A\in\mathcal{T}\\ \mathcal{T}\models r\sqsubseteq s}} \langle -1\rangle s^B \ \vee\\ & \bigvee_{\substack{\exists s.B\sqsubseteq A\in\mathcal{T}\\ \mathcal{T}\models u\sqsubseteq s}} \langle 1\rangle s_u^B\vee\ldots\vee\langle|\mathcal{T}|\rangle s_u^B\end{aligned}$$

For $l \in \Sigma_N$ with $\{r, b\} \subseteq l$ for a role r and $b \in \mathsf{C}_{\mathsf{core}}$, let:

$$\begin{aligned}\delta(s^A, l) = & \bigvee_{\mathcal{T} \models B_1 \sqcap \cdots \sqcap B_n \sqsubseteq A} \langle 0 \rangle s^{B_1} \wedge \ldots \wedge \langle 0 \rangle s^{B_n} \;\vee \\ & \bigvee_{\substack{\exists s^-.B \sqsubseteq A \in \mathcal{T} \\ \mathcal{T} \models r \sqsubseteq s}} \langle -1 \rangle s^B_b \;\vee \\ & \bigvee_{\substack{\exists s.B \sqsubseteq A \in \mathcal{T} \\ \mathcal{T} \models u \sqsubseteq s}} \langle 1 \rangle s^B_u \vee \ldots \vee \langle |\mathcal{T}| \rangle s^B_u\end{aligned}$$

All pairs $(s, l) \in S \times \Sigma_\varepsilon \uplus \Sigma_N$ that have not been mentioned yet will never occur in a run on a proper tree, so for those we just define $\delta(s, l) = \mathsf{false}$.

Correctness of $\mathfrak{A}_1^\lambda$. We now argue that $\mathfrak{A}_1^\lambda$ accepts a tree if and only if it encodes a tuple $(\mathcal{A}, \mathbf{a}, p)$ such that Condition 1 is fulfilled.

Let Condition 1 be fulfilled. We show that the automation accepts. This entails that the homomorphism pattern λ is compatible with the $\mathcal{B}$ and $\mathbf{a}$ that encoded in the label l at the root node of the tree. Hence, $\mathfrak{A}_1^\lambda$ does not reject immediately. Let h be the homomorphism from Condition 1. Since h is a homomorphism for all variables z of q_t that are mapped to the core of $\mathcal{A}$ and all atoms $A(z)$ from q_t, we have that $\mathcal{A}, \mathcal{T} \models A(h(z))$, so the conjunction (5.3) will succeed. For the second conjunction, consider a set of variables $Z \subseteq \mathsf{var}(q_t)$ described by the first conjunction when considering a core individual b. Let t be the set of concept names derived at b in the universal model of $\mathcal{T}$ and $\mathcal{A}$. We argue that that $(t, f) \in R$: By the definition of λ, the homomorphism h maps every variable from Z either to b or to the subtree below b. Since $\mathcal{T}$ is assumed to be in normal form, the subtree generated below b only depends on t and we can define g to be the restriction of h to Z. This function g witnesses that $(t, f) \in R$. For this set t, the last conjunction will of course succeed, since t was chosen to be the set of all concept names derived at b.

For the other direction, let (T, L) be a proper tree that is accepted by $\mathfrak{A}_1^\lambda$ and that encodes the tuple $(\mathcal{A}, \mathbf{a}, p)$. We need to construct the homomorphism h such that Condition 1 is fulfilled. Since the automaton does not reject immediately and the conjunction in (5.4) is fulfilled, there are $b_1, \ldots, b_n \in \mathsf{C}_{\mathsf{core}}$ such that the sets $Z_i = \lambda^{-1}(\{b_i\} \times \{\mathsf{core}, \mathsf{subtree}\}) \neq \emptyset$ form a partition of $\mathsf{var}(q_t)$ and for every i there is a type t_i derived at b_i in the universal model of $\mathcal{T}$ and $\mathcal{A}$ such that $(t_i, \pi_2 \circ \lambda|_{Z_i}) \in R$. The latter means that there is a homomorphism g_i from q_t restricted to Z_i to the tree that is generated below b_i in the universal model of $\mathcal{T}$ and $\mathcal{A}$. The homomorphism h is obtained by combining the g_i for all i. The second condition in the definition of Σ_ε guarantees that h also preserves roles between variables that lie in different Z_i.

Definition of $\mathfrak{A}_2$. The automaton $\mathfrak{A}_2$ checks Condition 2 by traversing the input tree once from the root to the leaves and guessing the homomorphism from p to q along the way. Some care is required since p is represented only implicitly in the input.

To define $\mathfrak{A}_2$, we precompute three relations R, R' and R''. Let R be a ternary relation between ABoxes $\mathcal{B}$ over $\mathsf{C}_{\mathsf{core}}$, disjuncts $d \in \mathbf{M}^-(\mathcal{B})$ and functions f from $\mathsf{ind}(\mathcal{B})$ to $\mathsf{var}(q)$. A triple $(\mathcal{B}, d, f)$ is in R if and only if there exists a homomorphism $h : d \to q$ such that $h(b) = f(b)$ for all $b \in \mathsf{ind}(\mathcal{B})$. Let R' be a binary relation between unary mappings from $\mathbf{M}$ and variables from q. A pair $(\varphi(x) \to A(x), y)$ is in R' if and only if there is a homomorphism $h : \varphi(x) \to q$ such that $h(x) = y$. Let R'' be a binary relation between binary mappings from $\mathbf{M}$ and pairs of variables from q. A triple $(\varphi(x, x') \to r(x, x'), y, y')$ is in R'' if there is a homomorphism $h : \varphi(x, x') \to q$ such that $h(x) = y$ and $h(x') = y'$. All three relations can be computed in ExpTime, since they all only check for homomorphisms between structures of polynomial size.

Let $\mathfrak{A}_2 = (S, \delta, \Sigma_\varepsilon \uplus \Sigma_N, s_0, c)$ where

$$S = \{s_0\} \uplus \{s^b_y \mid b \in \mathsf{C}_{\mathsf{core}} \text{ and } y \in \mathsf{var}(q)\} \\ \uplus \{s_y \mid y \in \mathsf{var}(q)\}$$

and $c(s) = 0$ for every $s \in S$, but the actual value of $c(s)$ does not matter since all runs of $\mathfrak{A}_2$ on proper trees will be finite.

For $l \in \Sigma_\varepsilon$ and $\mathcal{B} \in l$ and $d \in \mathbf{M}^-(\mathcal{B})$ the disjunct defined by the mappings in l, we define:

$$\delta(s_0, l) = \bigvee_{\substack{f:\mathsf{ind}(\mathcal{B})\to\mathsf{var}(q) \\ (\mathcal{B},d,f)\in R}} \bigwedge_{b\in\mathsf{ind}(\mathcal{B})} \bigwedge_{i\in\{1,\dots,|\mathsf{C}_{\mathsf{core}}|\cdot|\mathcal{T}|\}} [i]s^b_{f(b)}$$

For $l \in \Sigma_N$ we define:

$$\delta(s^b_y, l) = \begin{cases} \mathsf{true} & \text{if } b \notin l \\ \langle 0\rangle s_y & \text{if } b \in l \end{cases}$$

For $l = (\Theta, M, \mu) \in \Sigma_N$ and $y \in \mathsf{var}(q)$ let Y^y_l be the set of all $y' \in \mathsf{var}(q)$ such that $(\varphi(x, x') \to r(x, x'), y, y') \in R''$, where $\varphi(x, x') \to r(x, x')$ is the mapping M corresponding to the unique role in Θ, and such that $(\varphi(x) \to A(x), y') \in R'$ for every concept name $A \in \Theta$, where $\varphi(x) \to A(x)$ is the mapping $\mu(A)$. Then we define:

$$\delta(s_y, l) = \bigvee_{y'\in Y^y_l} \bigwedge_{i\in\{1,\dots,|\mathcal{T}|\}} [i]s_{y'}$$

Correctness of $\mathfrak{A}_2$. We now argue that $\mathfrak{A}_2$ accepts a tree (T, L) if and only if it encodes a tuple $(\mathcal{A}, \mathbf{a}, p)$ such that $p \to q$.

Assume there is homomorphism $h : p \to q$. We use h to describe a run of $\mathfrak{A}_2$ on (T, L) that traverses the tree once from the root to the leaves. At the root, the automaton chooses as f the restriction of h to $\mathsf{ind}(\mathcal{B})$, which is possible because $(\mathcal{B}, d, f) \in R$. If the run is in a configuration (t, s_y), where $t \in T$ corresponds to an individual a in $\mathcal{A}$, then we choose $h(a)$ as y'. Because h is a homomorphism, one can check that $y' \in Y^y_{L(t)}$. Thus, $\mathfrak{A}_2$ accepts (T, L).

For the other direction, let (T, L) be a tree encoding a tuple $(\mathcal{A}, \mathbf{a}, p)$ that is accepted by $\mathfrak{A}_2$. Let ρ be an accepting run of $\mathfrak{A}_2$ on (T, L). We obtain a homomorphism $h : p \to q$ by gluing together all of the following homomorphisms:

- The homomorphism from p restricted to facts generated by facts in $\mathcal{B}$ to q that exists by the choice of f in the root node.
- All homomorphisms h' obtained as follows: Consider any non-core fact α from $\mathcal{A}$. This fact appears in the label $L(t)$ of some t in (T, L). Since ρ is accepting, it will visit t in some state of the form s_y and chooses $y' \in Y^y_{L(t)}$. Because α is encoded in $L(t)$, it follows by the definition of $Y^y_{L(t)}$ that there is a homomorphism from the body of the mapping corresponding to α to q, which we choose as h'.

The information that is passed along in the states of the automaton guarantees that all these homomorphisms can be glued together to obtain a single homomorphism $h : p \longrightarrow q$.

5.7 Expressibility and Verification in $\mathcal{EL}$: Lower Bounds

We establish lower bounds that match the upper bounds obtained in the previous three sections and show that they even apply to $[\mathcal{EL}, \text{GAV}]$, that is, neither inverse roles nor role hierarchies are required.

Theorem 5.24.

1. *For $[\mathcal{EL}, GAV]$, the rUCQ-to-UCQ expressibility and verification problems are both coNExpTime-hard.*
2. *For $[\mathcal{EL}, GAV]$, the UCQ-to-UCQ expressibility and verification problems are both 2-ExpTime-hard*

By Theorem 5.8, it suffices to establish the lower bounds for the expressibility problem. We prove both points of Theorem 5.24 by a reduction from certain OMQ containment problems. For Point 1, we reduce from the following problem.

Theorem 5.25. *[Bie+16] Given $Q_1 = (\mathcal{T}, \Sigma, q_1)$ and $Q_2 = (\mathcal{T}, \Sigma, q_2)$ with $\mathcal{T}$ an $\mathcal{ELI}$-ontology, q_1 an AQ, and q_2 a rooted UCQ, it is coNExpTime-hard to decide whether $Q_1 \subseteq_\Sigma Q_2$, even when*

1. *$q_2(x)$ uses only symbols from Σ and*
2. *no symbol from Σ occurs on the right-hand side of a CI in $\mathcal{T}$.*

We first give some justification of Theorem 5.25. The hardness proof in [Bie+16] actually produces as q_2 a rooted CQ, but does not satisfy Condition 2. However, the only exception to Condition 2 are CIs of the form $D \sqsubseteq C_q$ where C_q is a specially crafted $\mathcal{ELI}$-concept that uses only symbols from Σ and is designed to 'make the query q_2 true', that is, whenever $d \in C_q^{\mathcal{I}}$ in an interpretation $\mathcal{I}$, then $\mathcal{I} \models q_2(d)$. It can be verified that the reduction in [Bie+16] still works when replacing the rooted CQ q_2 with the rooted UCQ $q_2 \vee \bigvee_{D \sqsubseteq C_q \in \mathcal{T}} q_D$, where q_D is the concept D viewed as a unary CQ in the obvious way, and then deleting all CIs of the form $D \sqsubseteq C_q$ from $\mathcal{T}$. Arguably, this modification even yields the more natural reduction, avoided in [Bie+16] to ensure that q_2 is a CQ rather than a UCQ.

To prove Point 1 of Theorem 5.24, we first establish it for $[\mathcal{ELI}, \text{GAV}]$ instead of for $[\mathcal{EL}, \text{GAV}]$ and in a second step show how to get rid of inverse roles. To reduce the containment problem in Theorem 5.25 to rUCQ-to-UCQ expressibility in $[\mathcal{ELI}, \text{GAV}]$, let $Q_1 = (\mathcal{T}, \Sigma, A_0(x))$ and $Q_2 = (\mathcal{T}, \Sigma, q)$ be as in that theorem. We define an OBDA-specification $\mathcal{S} = (\mathcal{T}', \mathbf{M}, \mathsf{S})$ and a query q_s over S as follows. Let B be a fresh concept name and define:

$$\begin{array}{rcl} \mathcal{T}' & = & \mathcal{T} \cup \{A_0 \sqsubseteq B\} \\ \mathsf{S} & = & \Sigma \cup \{B\} \\ q_s(x) & = & B(x) \vee q(x) \end{array}$$

Note that q_s is a rooted UCQ, as required. Moreover, the set $\mathbf{M}$ of mappings contains $A(x) \longrightarrow A(x)$ for all concept names $A \in \mathsf{S}$ and $r(x, y) \longrightarrow r(x, y)$ for all role names $r \in \mathsf{S}$, which means we have $q = \mathbf{M}(q)$ and $q = \mathbf{M}^{-}(q)$ for all S-queries q. Informally, the CI $A_0 \sqsubseteq B$ 'pollutes' B, potentially preventing the disjunct $B(x)$ of q_s to be expressible, but this is not a problem if (and only if) $Q_1 \subseteq Q_2$.

The following lemma establishes the correctness of the reduction and finishes the proof of Point 1 of Theorem 5.24 for $[\mathcal{ELI}, \text{GAV}]$ instead of $[\mathcal{EL}, \text{GAV}]$.

Lemma 5.26. *$Q_1 \subseteq Q_2$ if and only if q_s is UCQ-expressible in $\mathcal{S}$.*

In short, $Q_1 \not\subseteq Q_2$ if and only if there is a tree-shaped Σ-ABox witnessing this if and only if such an ABox, viewed as a CQ, is a disjunct of an infinitary UCQ-rewriting q_r of the OMQ $Q = (\mathcal{T}', \mathsf{S}, q_s)$ if and only if $q_r \not\subseteq q_s$. The latter is the case if and only if q_s is not UCQ-expressible in $\mathcal{S}$ by Theorem 5.8 and since $\mathbf{M}(q_s) = q_s$ and $\mathbf{M}^{-}(q_r) = q_r$.

Proof. Consider the OMQ $Q = (\mathcal{T}', \mathsf{S}, \mathbf{M}(q_s))$ and the infinitary UCQ q_r that consists of the following CQs:

1. $B(x)$,
2. each CQ from $q(x)$,
3. for every tree-shaped Σ-ABox $\mathcal{A}$ with root $a \in \text{cert}_{Q_1}(\mathcal{A})$, $(\mathcal{A}, a)$ viewed as a CQ.

It is easy to verify that q_r is a rewriting of Q. In particular, by Point 2 of Theorem 5.25, $q(x)$ uses only symbols from Σ which cannot be derived using the ontology and the restriction to tree-shaped ABoxes in Point 3 is sanctioned by Lemma 2.3.

"if". Assume that $Q_1 \not\subseteq Q_2$. We need to show that q_s is not UCQ-expressible. By Theorem 5.8, it is sufficient to show $q_r \not\subseteq q_s$. Since $Q_1 \not\subseteq Q_2$, there exists a pair $(\mathcal{A}, a)$ such that $a \in \text{cert}_{Q_1}(\mathcal{A})$ and $a \notin \text{cert}_{Q_2}(\mathcal{A})$. By Lemma 2.3 and as already observed in [Bie+16], we can assume that $\mathcal{A}$ is tree-shaped with root a. Furthermore, we can assume that B does not occur in $\mathcal{A}$: Since B is a fresh concept name, no B-assertion is needed to derive $A_0(a)$. Let p be $(\mathcal{A}, a)$ viewed as a CQ. Clearly, $a \in \text{ans}_{q_r}(\mathcal{A})$, since p is a disjunct in q_r. On the other hand, $a \notin \text{ans}_{q_s}(\mathcal{A})$: From none of the CQs in the UCQ q_s there is a homomorphism to p. This is true for $B(x)$, since B does not occur in p. It is also true for the disjuncts of $q(x)$, since $a \notin \mathsf{cert}_{Q_2}(\mathcal{A})$ and q does not use symbols that occur on the right-hand side of CIs in $\mathcal{T}$.

"only if". Assume that $Q_1 \subseteq Q_2$. We have to show that $q_r \subseteq_{\mathbf{S}} q_s$, or in other words, that for every CQ p in q_r, there is a CQ $p' \in q_s$ with $p' \to p$. This is clear for the CQs from Points 1 and 2 of the construction of q_r, since all of them appear as a CQ also in $q_s(x)$. For Point 3, let p be obtained from a pair $(\mathcal{A}, a)$ for a tree-shaped Σ-ABox with root $a \in \mathsf{cert}_{Q_1}(\mathcal{A})$. From $Q_1 \subseteq Q_2$, we obtain $a \in \mathsf{cert}_{Q_2}(\mathcal{A})$. With Point 1 and 2 of Theorem 5.25, this yields $a \in \mathsf{ans}_q(\mathcal{A})$, so there exists a disjunct p' of q such that $p' \to p$, as required. □

Now we describe how to replace the $\mathcal{ELI}$-ontology $\mathcal{T}$ with an $\mathcal{EL}$-ontology. The crucial observation is that the hardness proof from [Bie+16] uses only a single symmetric role S implemented as a composition $r_0^-; r_0$ with r_0 a role name, and that it is possible to replace this composition with a normal role name r in $\mathcal{T}$ when reintroducing it in $\mathbf{M}^-(q_r)$ via mappings $r_0(x,y) \wedge r_0(x,z) \to r(y,z)$ where q_r is an infinitary UCQ-rewriting of the OMQ Q mentioned above.

It can be verified that query Q_1 from Theorem 5.25 is 'one-way', that is, $\mathcal{T}$ verifies the existence of a (homomorphic image of a) certain tree-shaped sub-ABox *from the bottom up* and then Q_1 makes $q_1 = A_0(x)$ true at the root when the tree-shaped ABox was found. This one-way behaviour can be made formal in terms of derivations of A_0 by $\mathcal{T}$, introduced in Section 2.8.

If $\mathcal{A}$ is tree-shaped with root a and (T, V) is a derivation tree for the fact $A_0(a)$, then we say that (T, V) is *bottom-up* if the following condition is satisfied: if y is a successor of x in T, $V(x) = (a_x, A_x)$, and $V(y) = (a_y, A_y)$, then $a_x = a_y$ or a_y is a successor (but not a predecessor) of a_x in $\mathcal{A}$, that is, a_y is further away from the root of $\mathcal{A}$ than a_x is. The OMQ Q_1 is one-way in the sense that if $\mathcal{A}$ is a tree-shaped Σ-ABox with root $a \in \mathsf{cert}_{Q_1}(\mathcal{A})$, then all derivations of $A_0(a)$ in $\mathcal{A}$ are bottom-up. Note that a corresponding statement for Q_2 makes little sense since by Conditions 1 and 2 of Theorem 5.25, answers to Q_2 on an ABox $\mathcal{A}$ are independent of $\mathcal{T}$.

We can exploit the one-way property of Q_1 as follows. In the hardness proof in [Bie+16], all involved ontologies, signatures, and queries use only a single role name S that is interpreted as a *symmetric role*, and in fact represented via the composition $r_0^-; r_0$ where r_0 is a fixed 'standard' (non-symmetric) role name. We can replace S with a standard role name r in $\mathcal{T}$ and in Σ, turning the $\mathcal{ELI}$-ontology $\mathcal{T}$ into an $\mathcal{EL}$-ontology. The mapping $r_0(x,y) \to r_0(x,y)$ from $\mathbf{M}$ in the original reduction is then replaced with $r_0(x,y) \wedge r_0(x,z) \to r(y,z)$; note that, in q_s, we keep the composition $r_0^-; r_0$ from the original reduction. We claim that, again, the following holds.

Lemma 5.27. *$Q_1 \subseteq Q_2$ if and only if q_s is UCQ-expressible in $\mathcal{S}$.*

Proof. (sketch) Let the UCQ q_r be defined as before except that the CQs from q are replaced with those from $\mathbf{M}(q)$. It can be verified that q_r is a rewriting of $Q = (\mathcal{T}', \mathbf{S}, \mathbf{M}(q_s))$. Moreover, it is easy to see that $\mathbf{M}^-(\mathbf{M}(q)) = q$. This and the fact that Q_1 is one-way can be used to verify that $\mathbf{M}^-(q_r)$ is identical to the query q_r from the original reduction (the one-way property is needed to see that the CQs from Point 3 of the definition of q_r are identical in both cases, except that S is replaced with r). From there, the proof proceeds as in Lemma 5.26 to show that $Q_1 \subseteq Q_2$ if and only if for every CQ p in $\mathbf{M}^-(q_r)$, there is a CQ $p' \in q_s$ with $p' \to p$. □

Next, we prove Point 2 of Theorem 5.24. We identify a suitable containment problem proved 2-ExpTime in [Bie+16] and then proceed very similarly to the case of Point 1.

Theorem 5.28. *[Bie+16] Containment between OMQs $Q_1 = (\mathcal{T}, \Sigma, q_1)$ and $Q_2 = (\mathcal{T}, \Sigma, q_2)$ with $\mathcal{T}$ an $\mathcal{ELI}$-ontology, q_1 of the form $\exists x\, A_0(x)$, and q_2 a UCQ is 2-ExpTime-hard even when*

1. *$q_2(x)$ uses only symbols from Σ and*
2. *no symbol from Σ occurs on the right-hand side of a CI in $\mathcal{T}$.*

Again, the actual hardness proof from [Bie+16] needs to be slightly modified to actually achieve what is stated in Theorem 5.28. In particular, CIs of the form $D \sqsubseteq C_q$ again have to be turned into additional disjuncts of the UCQ q_2. This requires an additional modification of the reduction since there are CIs of the form $D \sqsubseteq C_q$ where D uses symbols that are not from Σ and occur on the right-hand side of CIs in $\mathcal{T}$. In a nutshell and speaking in terms of the notation from [Bie+16], the concept names H and W' need to be added to Σ and their presence at the intended places must be 'verified in the input' rather than 'enforced by the ontology'. After this is done, the only (minor) problem remaining is that the concept name G is used (exactly twice) in a (single) CQ p in q_2, but it does occur on the right-hand side of two CIs which are $G_1 \sqsubseteq G$ and $G_2 \sqsubseteq G$. Here, G_1, G_2 are from Σ and do not occur on the right-hand side of a CI. This can be fixed by replacing p with four CQs in the UCQ q_2, replacing the two occurrences of G with G_1 or G_2 in all possible ways.

Point 2 of Theorem 5.24 is again first proved for $[\mathcal{ELI}, \text{GAV}]$ instead of for $[\mathcal{EL}, \text{GAV}]$. This is done by reduction from the containment problem in Theorem 5.28. Let $Q_1 = (\mathcal{T}, \Sigma, \exists x\, A_0(x))$ and $Q_2 = (\mathcal{T}, \Sigma, q)$. We define an OBDA-specification $\mathcal{S} = (\mathcal{T}', \mathbf{M}, \mathbf{S})$ and a query $q_s()$ over $\mathbf{S}$ as follows. Let B be a fresh concept name and define:

$$\begin{aligned}
\mathcal{T}' &= \mathcal{T} \cup \{A_0 \sqsubseteq B\} \\
\mathbf{S} &= \Sigma \cup \{B\} \\
q_s() &= \exists x\, B(x) \vee q()
\end{aligned}$$

The set $\mathbf{M}$ of mappings again contains $A(x) \to A(x)$ for all concept names $A \in \mathbf{S}$ and $r(x,y) \to r(x,y)$ for all role names $r \in \mathbf{S}$. The proof of correctness is essentially identical to the the proof of Lemma 5.26.

The approach to eliminating inverse roles is also exactly identical to the coNExpTime case. In fact, the OMQ $(\mathcal{T}, \Sigma, A_0(x))$ is again one-way and all involved ontologies, signatures and queries again only use the single symmetric role name S represented as the composition $r_0^-; r_0$. Consequently, the same arguments apply.

5.8 Conclusion

We believe that several interesting questions remain. For example, our lower bounds only apply when the source query is a UCQ and it would be interesting to see whether the complexity drops when source queries are CQs. It would also be interesting to consider ontologies formulated in more expressive DLs such as $\mathcal{ALC}$. As a first observation in this direction, we note the following undecidability result, where $\mathcal{ALCF}$ is $\mathcal{ALC}$ extended with (globally) functional roles.

Theorem 5.29. *In $[\mathcal{ALCF}, GAV]$, the AQ-to-$\mathcal{Q}$ expressibility and verification problems are undecidable for any $\mathcal{Q} \in \{AQ, CQ, UCQ\}$.*

Proof. We provide a reduction from the emptiness of AQs w.r.t. $\mathcal{ALCF}$-ontologies, which is undecidable [Baa+16]. Let $(\mathcal{T}, \mathbf{S}, A_0)$ be an OMQ with $\mathcal{T}$ in $\mathcal{ALCF}$ and $A_0(x)$ an AQ. Let B_0 be a fresh concept name and define an OBDA-specification $\mathcal{S} = (\mathcal{T}', \mathbf{M}, \mathbf{S}')$ where $\mathcal{T}' = \mathcal{T} \cup \{A_0 \sqsubseteq B_0\}$, $\mathbf{S}' = \mathbf{S} \cup \{B_0\}$, and $\mathbf{M}$ consists of the mappings $A(x) \longrightarrow A(x)$ for all concept names A in $\mathbf{S}'$ and $r(x, y) \longrightarrow r(x, y)$ for all role names r in $\mathbf{S}'$. We consider expressibility of the AQ $B_0(x)$. In fact, it is possible to verify the following:

1. if A_0 is empty w.r.t. $\mathcal{T}$, then $B_0(x)$ is a realization of $B_0(x)$ in $\mathcal{S}$;
2. if A_0 is non-empty w.r.t. $\mathcal{T}$, then there is a $\mathbf{S}$-database D and a constant $a \in \mathsf{adom}(D)$ such that $a \in \mathsf{cert}_Q(D)$. Define the $\mathbf{S}'$-database $D' = D \cup \{B_0(a)\}$. Now, $B_0(x)$ is not determined in $\mathcal{S}$ in the sense that $a \in \mathsf{ans}_{B_0}(D') \setminus \mathsf{ans}_{B_0}(D)$ but every model of $\mathbf{M}(D)$ and $\mathcal{T}'$ is also a model of $\mathbf{M}(D')$ and $\mathcal{T}'$, and vice versa. Consequently, $B_0(x)$ is not $\mathcal{Q}$-expressible in $\mathcal{S}$ for any $\mathcal{Q} \in \{\mathrm{AQ}, \mathrm{CQ}, \mathrm{UCQ}\}$ or, in fact, any other query language.

□

Regarding the expressibility problem, we note that the realization $\mathbf{M}(q_s)$ identified by Theorem 5.8 does not use any symbols introduced by the ontology and, in fact, is also a realization regarding the empty ontology. It would be interesting to understand how to obtain realizations that make better use of the ontology and to study setups where it can be unavoidable to exploit the ontology in realizations. This is the case, for example, when source queries are formulated in Datalog, the ontology is formulated in (some extension of) $\mathcal{EL}$, and target queries are UCQs. Finally, we note that it would be natural to study maximally contained realizations instead of exact ones and to take into account constraints over the source databases.

6 Conclusion

In this thesis, we have investigated the complexity of several problems in the field of ontology-mediated querying with Horn-DL ontologies. We summarize our results and put them into context.

- The objective of Chapter 3 was to deepen the understanding of fine-grained data complexity and rewritability questions about OMQs. We have classified data complexity of OMQ from $(\mathcal{EL}, \mathrm{CQ})$ in the most fine-grained way. We showed that every such OMQ is either in AC^0 or NL-complete or PTime-complete. We also showed that containment in NL coincides with rewritability into linear Datalog and that there is not constant bound on the width of the rewritings. Furthermore, we gave characterizations for the different cases and used these to show that all related the meta problems, like deciding linear Datalog rewritability, is ExpTime-complete.

  We think these results are an important step towards the formulated goal. We notice the proofs in Chapter 3 are rather technical, even though we deal with the rather simple description logic $\mathcal{EL}$. The observations from Section 3.8 suggest that going to $\mathcal{ELI}$ would be a step into the territory of long standing open questions in the world of CSPs.

- In Chapter 4, we initiated the research on query-by-example and the query definability problem in the presence of ontologies formulated in Horn-DLs. We showed that finding a distinguishing CQ is coNExpTime-complete for ontologies in Horn-$\mathcal{ALC}$ and undecidable for ontologies in Horn-$\mathcal{ALCI}$, even for ontologies in $\mathcal{ELI}$. We also determined tight upper and lower bounds on the size of witness queries in the Horn-$\mathcal{ALC}$ case.

  The goal of this chapter was to improve usability of ontology enriched systems and to understand what difference an ontology can make in the query-by-example problem. The undecidability result may sound disappointing regarding usability, but it points out an interesting category of undecidable problems that have to do with products of universal models in the presence of inverse roles, which is interesting from a theoretical perspective. On the practical side, this result suggests that considering less expressive DLs like the DL-Lite family is a good direction to pursue.

- In Chapter 5, we investigated the expressibility problem and the verification problem for queries in the OBDA context with Horn DLs. We proved that these problems are Π_2^p-complete DL-Lite and all weaker ontology languages. For ontology languages between $\mathcal{EL}$ and $\mathcal{ELHI}$, we proved coNExpTime-complete for rooted source queries and 2-ExpTime-completeness for unrestricted queries.

Like Chapter 4, the objective of this chapter was also to improve usability of ontologies in practice, and in fact, the Π_2^p-completeness results show that in the OBDA setting, DL-Lite ontologies can be added in the data integration setting without having a negative effect on the complexity, compared to the situation without ontologies.

We already addressed concrete open questions and opportunities for future work in the main chapters, so we confine ourselves to more general remarks here. In this thesis, we only considered a few reasoning problems in combination with a few description logics. There is still a huge number of interesting combinations that have not been considered or fully understood yet. Also, the popularity of ontologies gives rise to new reasoning problems over time, so we think there are still a lot of interesting questions waiting to be solved in this field.

One thing we remark regarding all chapters of the thesis has to do with inverse roles. It seems like a useful feature for designing ontologies to add inverse roles, but we have seen that adding inverse roles to an ontology language can have an immense effect of the complexity of a reasoning problem, even in two ways: First, we have seen examples where the computational complexity of a reasoning problem increases when inverse roles are added. This effect has been known before, consider for example the subsumption problem in $\mathcal{EL}$ and in $\mathcal{ELI}$, where the complexity jumps from PTime to ExpTime [BBL05]. But secondly, inverse roles can have complex and unforeseen effects that make these problems technically harder to tackle. Of course, these two effects often go hand in hand. We suspect there might be more things to discover and to understand regarding inverse roles, things that might help to solve the open problem of Chapter 3, that is, a complete classification of data complexity of OMQs from $(\mathcal{ELI}, \text{CQ})$.

Of all the open questions in this thesis, the classification of data complexity of OMQs from $(\mathcal{ELI}, \text{CQ})$ is definitely the most intriguing to us. Perhaps it is also the most difficult one, since the result would have implications on open problems in the CSP world. In fact, classifying these OMQs corresponds to classifying CSPs with tree obstructions, and the model-theoretic characterizations and questions like rewritability into symmetric Datalog are theoretically interesting, as explained in Section 3.8.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009. ISBN: 978-0-521-42426-4.

[ADK16] Marcelo Arenas, Gonzalo I. Diaz, and Egor V. Kostylev. “Reverse Engineering SPARQL Queries”. In: *Proc. of WWW.* 2016, pp. 239–249.

[Afr11] Foto N. Afrati. “Determinacy and query rewriting for conjunctive queries and views”. In: *Theor. Comput. Sci.* 412.11 (2011), pp. 1005–1021.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995. ISBN: 0-201-53771-0.

[All+09] Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. “The complexity of satisfiability problems: Refining Schaefer’s theorem”. In: *J. Comput. Syst. Sci.* 75.4 (2009), pp. 245–254.

[Ang87] Dana Angluin. “Queries and Concept Learning”. In: *Machine Learning* 2.4 (1987), pp. 319–342.

[Art+09] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. “The DL-Lite Family and Relations”. In: *J. Artif. Intell. Res.* 36 (2009), pp. 1–69.

[Baa+07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications.* 2nd. Cambridge Univ. Press, 2007.

[Baa+16] Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. “Query and Predicate Emptiness in Ontology-Based Data Access”. In: *J. Artif. Intell. Res.* 56 (2016), pp. 1–59.

[Baa+17] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logics.* Cambride University Press, 2017.

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. “Pushing the $\mathcal{EL}$ Envelope”. In: *Proc. of IJCAI.* 2005, pp. 364–369.

[BCL15] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. “Learning Path Queries on Graph Databases”. In: *Proc. of EDBT.* 2015, pp. 109–120.

[BCS14] Angela Bonifati, Radu Ciucanu, and Slawek Staworko. “Interactive Inference of Join Queries”. In: *Proc. of EDBT.* 2014, pp. 451–462.

[BCS16] Angela Bonifati, Radu Ciucanu, and Slawek Staworko. “Learning Join Queries from User Examples”. In: *ACM Trans. Database Syst.* 40.4 (2016), 24:1–24:38.

[Bie+14] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. "Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP". In: *ACM Trans. Database Syst.* 39.4 (2014), 33:1–33:44.

[Bie+16] Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. "First Order-Rewritability and Containment of Conjunctive Queries in Horn Description Logics". In: *Proc. of IJCAI.* 2016, pp. 965–971.

[BKL08] Andrei A. Bulatov, Andrei A. Krokhin, and Benoit Larose. "Dualities for Constraint Satisfaction Problems". In: *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar].* Vol. 5250. LNCS. Springer, 2008, pp. 93–124.

[BLB08] Franz Baader, Carsten Lutz, and Sebastian Brandt. "Pushing the EL Envelope Further". In: *Proc. of OWLED Workshop on OWL.* Vol. 496. CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[BLR97] Catriel Beeri, Alon Y. Levy, and Marie-Christine Rousset. "Rewriting Queries Using Views in Description Logics". In: *Proc. of PODS.* ACM Press, 1997, pp. 99–108.

[BLW13] Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. "First Order-Rewritability of Atomic Queries in Horn Description Logics". In: *Proc. of IJCAI.* IJCAI/AAAI, 2013, pp. 754–760.

[BO15] Meghyn Bienvenu and Magdalena Ortiz. "Ontology-Mediated Query Answering with Data-Tractable Description Logics". In: *Proc. of Reasoning Web.* Vol. 9203. LNCS. Springer, 2015, pp. 218–307.

[Bod98] Hans L. Bodlaender. "A Partial k-Arboretum of Graphs with Bounded Treewidth". In: *Theor. Comput. Sci.* 209.1-2 (1998), pp. 1–45.

[Bon+14] Angela Bonifati, Radu Ciucanu, Aurélien Lemay, and Slawek Staworko. "A Paradigm for Learning Queries on Big Data". In: *Proc. of Data4U@VLDB.* 2014, p. 7.

[Bot+16] Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. "Inseparability and Conservative Extensions of Description Logic Ontologies: A Survey". In: *Proc. of RW.* 2016.

[Bot+19] Elena Botoeva, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. "Query inseparability for ALC ontologies". In: *Artif. Intell.* 272 (2019), pp. 1–51.

[BR17] Pablo Barceló and Miguel Romero. "The Complexity of Reverse Engineering Problems for Conjunctive Queries". In: *Proc. of ICDT.* 2017.

[BST07] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. "Computing the least common subsumer w.r.t. a background terminology". In: *J. Applied Logic* 5.3 (2007), pp. 392–420.

[Bul17] Andrei A. Bulatov. "A Dichotomy Theorem for Nonuniform CSPs". In: *Proc. of FOCS.* 2017, pp. 319–330.

[Cal+02] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. “Lossless Regular Views”. In: *Proc. of PODS.* ACM, 2002, pp. 247–258.

[Cal+09] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. “Ontologies and Databases: The DL-Lite Approach”. In: *Proc. of Reasoning Web.* 2009, pp. 255–356.

[Cal+12] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. “View-based query answering in Description Logics: Semantics and complexity”. In: *J. Comput. Syst. Sci.* 78.1 (2012), pp. 26–46.

[Cal+13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. “Data complexity of query answering in description logics”. In: *Artif. Intell.* 195 (2013), pp. 335–360.

[Cal+16] Diego Calvanese, Pietro Liuzzo, Alessandro Mosca, José Remesal, Martin Rezk, and Guillem Rull. “Ontology-based data integration in EPNet: Production and distribution of food during the Roman Empire”. In: *Eng. Appl. of AI* 51 (2016), pp. 212–229.

[CD15] Balder ten Cate and Víctor Dalmau. “The Product Homomorphism Problem and Applications”. In: *Proc. of ICDT.* 2015, pp. 161–176.

[CDK10] Catarina Carvalho, Víctor Dalmau, and Andrei A. Krokhin. “CSP duality and trees of bounded pathwidth”. In: *Theor. Comput. Sci.* 411.34-36 (2010), pp. 3188–3208.

[CDL00] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. “Answering Queries Using Views over Description Logics Knowledge Bases”. In: *Proc. of IAAI.* AAAI Press / The MIT Press, 2000, pp. 386–391.

[CGP12] Andrea Calì, Georg Gottlob, and Andreas Pieris. “Towards more expressive ontology languages: The query answering problem”. In: *Artif. Intell.* 193 (2012), pp. 87–128.

[Cim17] Gianluca Cima. “Preliminary Results on Ontology-based Open Data Publishing”. In: *Proc. of DL.* Vol. 1879. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[Col+16] Simona Colucci, Francesco M. Donini, Silvia Giannini, and Eugenio Di Sciascio. “Defining and computing Least Common Subsumers in RDF”. In: *J. Web Semant.* 39 (2016), pp. 62–80.

[CP95] William W. Cohen and C. David Page. “Polynomial Learnability and Inductive Logic Programming: Methods and Results”. In: *New Generation Comput.* 13.3&4 (1995), pp. 369–409.

[DAB16] Gonzalo I. Diaz, Marcelo Arenas, and Michael Benedikt. “SPARQLByE: Querying RDF data by example”. In: *PVLDB* 9.13 (2016), pp. 1533–1536.

[Dal05] Víctor Dalmau. “Linear datalog and bounded path duality of relational structures”. In: *Logical Methods in Computer Science* 1.1 (2005).

[DG97] Oliver M. Duschka and Michael R. Genesereth. “Answering Recursive Queries Using Views”. In: *Proc. of PODS.* ACM Press, 1997, pp. 109–116.

[DK08] Víctor Dalmau and Andrei A. Krokhin. “Majority constraints have bounded pathwidth duality”. In: *Eur. J. Comb.* 29.4 (2008), pp. 821–837.

[Eit+08] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Simkus. “Query Answering in the Description Logic Horn-SHIQ”. In: *Proc. of JELIA.* 2008, pp. 166–179.

[Eit+12] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. “Query Rewriting for Horn-SHIQ Plus Rules”. In: *Proc. of AAAI.* AAAI Press, 2012.

[ELT07] László Egri, Benoit Larose, and Pascal Tesson. “Symmetric Datalog and Constraint Satisfaction Problems in LogSpace”. In: *Electronic Colloquium on Computational Complexity (ECCC)* 14.024 (2007), p. 1.

[ELT08] László Egri, Benoit Larose, and Pascal Tesson. “Directed st-Connectivity Is Not Expressible in Symmetric Datalog”. In: *Proc. of ICALP.* Vol. 5126. LNCS. Springer, 2008, pp. 172–183.

[Fan+18] Nicola Fanizzi, Giuseppe Rizzo, Claudia d’Amato, and Floriana Esposito. “DLFoil: Class Expression Learning Revisited”. In: *Proc. of EKAW.* 2018, pp. 98–113.

[FKL19] Cristina Feier, Antti Kuusisto, and Carsten Lutz. “Rewritability in Monadic Disjunctive Datalog, MMSNP, and Expressive Description Logics”. In: *Logical Methods in Computer Science* (2019).

[FSS81] Merrick L. Furst, James B. Saxe, and Michael Sipser. “Parity, Circuits, and the Polynomial-Time Hierarchy”. In: *Proc. of FOCS.* 1981, pp. 260–270.

[Fun+19] Maurice Funk, Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. “Learning Description Logic Concepts: When can Positive and Negative Examples be Separated”. In: *Proc. of IJCAI.* 2019.

[Fun19] Maurice Funk. “Concept-By-Example in EL Knowledge Bases”. MA thesis. University of Bremen, 2019.

[FV98] Tomás Feder and Moshe Y. Vardi. “The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory”. In: *SIAM J. Comput.* 28.1 (1998), pp. 57–104.

[GJS18a] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. “Query-by-Example for Expressive Horn Description Logics”. In: *Proceedings of DL-workshop.* 2018.

[GJS18b] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. “Reverse Engineering Queries in Ontology-Enriched Systems: The Case of Expressive Horn Description Logic Ontologies”. In: *Proceedings of IJCAI.* 2018, pp. 1847–1853.

[GO14] Viliam Geffert and Alexander Okhotin. “Transforming Two-Way Alternating Finite Automata to One-Way Nondeterministic Automata”. In: *Proc. of MFCS.* 2014, pp. 291–302.

[Got+14] Georg Gottlob, Stanislav Kikot, Roman Kontchakov, Vladimir V. Podolskii, Thomas Schwentick, and Michael Zakharyaschev. “The price of query rewriting in ontology-based data access”. In: *Artif. Intell.* 213 (2014), pp. 42–59.

[GR17] Martin Grohe and Martin Ritzert. “Learning first-order definable concepts over structures of small degree”. In: *Proc. of LICS.* 2017, pp. 1–12.

[Han+15] Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. “Efficient Query Rewriting in the Description Logic $\mathcal{EL}$ and Beyond”. In: *Proc. of IJCAI.* AAAI Press, 2015, pp. 3034–3040.

[HM05] Peter Haase and Boris Motik. “A mapping system for the integration of OWL-DL ontologies”. In: *Proc. of IHIS’05.* ACM, 2005, pp. 9–16.

[HMS05] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. “Data Complexity of Reasoning in Very Expressive Description Logics”. In: *Proc. of IJCAI.* Professional Book Center, 2005, pp. 466–471.

[Hov+17] Dag Hovland, Roman Kontchakov, Martin G. Skjæveland, Arild Waaler, and Michael Zakharyaschev. “Ontology-Based Data Access to Slegge”. In: *Proc. of ISWC.* 2017, pp. 120–129.

[Imm99] Neil Immerman. *Descriptive complexity.* Graduate texts in computer science. Springer, 1999. ISBN: 978-1-4612-6809-3.

[Jim+15] Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, Ian Horrocks, Christoph Pinkel, Martin G. Skjæveland, Evgenij Thorstensen, and José Mora. “BootOX: Practical Mapping of RDBs to OWL 2”. In: *Proc. of ISWC.* Vol. 9367. LNCS. Springer, 2015, pp. 113–132.

[Kaz15] Alexandr Kazda. “*n*-permutability and linear Datalog implies symmetric Datalog”. In: *CoRR* abs/1508.05766 (2015).

[Kha+15] Evgeny Kharlamov, Dag Hovland, Ernesto Jiménez-Ruiz, Davide Lanti, Hallstein Lie, Christoph Pinkel, Martín Rezk, Martin G. Skjæveland, Evgenij Thorstensen, Guohui Xiao, Dmitriy Zheleznyakov, and Ian Horrocks. “Ontology Based Access to Exploration Data at Statoil”. In: *Proc. of ISWC.* Vol. 9367. LNCS. Springer, 2015, pp. 93–112.

[Kie02] Jörg-Uwe Kietz. “Learnability of Description Logic Programs”. In: *Proc. of ILP.* 2002, pp. 117–132.

[KL07] Adila Krisnadhi and Carsten Lutz. “Data Complexity in the $\mathcal{EL}$ family of Description Logics”. In: *Proc. of LPAR.* Vol. 4790. LNAI. Springer, 2007, pp. 333–347.

[KNG14] Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau. “Datalog Rewritability of Disjunctive Datalog Programs and its Applications to Ontology Reasoning”. In: *Proc. of AAAI.* AAAI Press, 2014, pp. 1077–1083.

[Kon+17] Boris Konev, Carsten Lutz, Ana Ozaki, and Frank Wolter. “Exact Learning of Lightweight Description Logic Ontologies”. In: *Journal of Machine Learning Research* 18 (2017), 201:1–201:63.

[KRH13] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. “Complexities of Horn Description Logics”. In: *ACM Trans. Comput. Logic* 14.1 (2013), 2:1–2:36.

[Lem+17] Domenico Lembo, Riccardo Rosati, Valerio Santarelli, Domenico Fabio Savo, and Evgenij Thorstensen. “Mapping Repair in Ontology-based Data Access Evolving Systems”. In: *Proc. of IJCAI.* ijcai.org, 2017, pp. 1160–1166.

[Lev+95] Alon Y Levy, Alberto O Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. “Answering Queries Using Views”. In: *Proc of PODS.* 1995, pp. 95–104.

[LH10] Jens Lehmann and Pascal Hitzler. “Concept learning in description logics using refinement operators”. In: *Machine Learning* 78.1-2 (2010), pp. 203–250.

[Lis12] Francesca A. Lisi. “A Formal Characterization of Concept Learning in Description Logics”. In: *Proc. of DL Workshop.* 2012.

[LMS18] Carsten Lutz, Johannes Marti, and Leif Sabellek. “Query Expressibility and Verification in Ontology-Based Data Access”. In: *Proceedings of KR.* AAAI Press, 2018, pp. 389–398.

[LS17a] Carsten Lutz and Leif Sabellek. “Ontology-Mediated Querying with EL: Trichotomy and Linear Datalog Rewritability”. In: *Proceedings of DL-workshop.* 2017.

[LS17b] Carsten Lutz and Leif Sabellek. “Ontology-Mediated Querying with the Description Logic EL: Trichotomy and Linear Datalog Rewritability”. In: *Proc. of IJCAI.* 2017, pp. 1181–1187.

[LS19] Carsten Lutz and Leif Sabellek. “A Complete Classification of the Complexity and Rewritability of Ontology-Mediated Queries based on the Description Logic EL”. submitted to AI Journal. 2019.

[LSW15] Carsten Lutz, Inanç Seylan, and Frank Wolter. “Ontology-Mediated Queries with Closed Predicates”. In: *Proc. of IJCAI.* AAAI Press, 2015, pp. 3120–3126.

[LT09] Benoit Larose and Pascal Tesson. “Universal algebra and hardness results for constraint satisfaction problems”. In: *Theor. Comput. Sci.* 410.18 (2009), pp. 1629–1647.

[Lut08] Carsten Lutz. “The Complexity of Conjunctive Query Answering in Expressive Description Logics”. In: *Proc. of IJCAR2008.* Vol. 5195. LNCS. Springer, 2008, pp. 179–193.

[LW12] Carsten Lutz and Frank Wolter. “Non-Uniform Data Complexity of Query Answering in Description Logics”. In: *Proc. of KR.* AAAI Press, 2012.

[LW17] Carsten Lutz and Frank Wolter. “The Data Complexity of Description Logic Ontologies”. In: *Logical Methods in Computer Science* 13.4 (2017).

[Mot+16] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. "Exemplar queries: a new way of searching". In: *VLDB J.* 25.6 (2016), pp. 741–765.

[Mot+17] Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. "New Trends on Exploratory Methods for Data Analytics". In: *PVLDB* 10.12 (2017), pp. 1977–1980.

[NSV10] Alan Nash, Luc Segoufin, and Victor Vianu. "Views and queries: Determinacy and rewriting". In: *ACM Trans. Database Syst.* 35.3 (2010), 21:1–21:41.

[NW97] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf, eds. *Foundations of Inductive Logic Programming.* Vol. 1228. LNCS. Springer, 1997.

[OEI] The On-Line Encyclopedia of Integer Sequences OEIS Foundation Inc. (2019). *Maximal number of regions obtained by joining n points around a circle by straight lines. Also number of regions in 4-space formed by n-1 hyperplanes.* URL: https://oeis.org/A000127 (visited on 05/03/2019).

[Pin+18] Christoph Pinkel, Carsten Binnig, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Wolfgang May, Andriy Nikolov, Ana Sasa Bastinos, Martin G. Skjæveland, Alessandro Solimando, Mohsen Taheriyan, Christian Heupel, and Ian Horrocks. "RODI: Benchmarking relational-to-ontology mapping generation quality". In: *Semantic Web* 9.1 (2018), pp. 25–52.

[Plo70] Gordon D. Plotkin. "A Note on Inductive Generalization". In: *Machine Intelligence* 5 (1970), pp. 153–163.

[PMH10] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. "Tractable query answering and rewriting under description logic constraints". In: *Journal of Applied Logic* 8.2 (2010), pp. 186–209.

[Pog+08] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Linking Data to Ontologies". In: *Journal on Data Semantics* 10 (2008), pp. 133–173.

[Rei04] Omer Reingold. "Undirected ST-Connectivity in Log-Space". In: *Electronic Colloquium on Computational Complexity (ECCC)* 094 (2004).

[Rei77] Raymond Reiter. "On Closed World Data Bases". In: *Proc. of Symposium on Logic and Data Bases.* Advances in Data Base Theory. Plemum Press, 1977, pp. 55–76.

[RKZ13] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. "Ontology-Based Data Access: Ontop of Databases". In: *Proc. of ISWC.* 2013, pp. 558–573.

[Ros07] Riccardo Rosati. "The Limits of Querying Ontologies". In: *Proc. of ICDT.* Vol. 4353. LNCS. Springer, 2007, pp. 164–178.

[Sch89] Petra Scheffler. *Die Baumweite von Graphen als ein Mass für die Kompliziertheit algorithmischer Probleme.* Report (Karl-Weierstrass-Institut für Mathematik). Akademie der Wissenschaften der DDR, Karl-Weierstrass-Institut für Mathematik, 1989.

[SM17] Juan F. Sequeda and Daniel P. Miranker. "A Pay-As-You-Go Methodology for Ontology-Based Data Access". In: *IEEE Internet Computing* 21.2 (2017), pp. 92–96.

[TCP14] Quoc Trung Tran, Chee Yong Chan, and Srinivasan Parthasarathy. "Query reverse engineering". In: *VLDB J.* 23.5 (2014), pp. 721–746.

[TLN99] Stefano Trisolini, Maurizio Lenzerini, and Daniele Nardi. "Data Integration and Warehousing in Telecom Italia". In: *Proc. of SIGMOD.* ACM Press, 1999, pp. 538–539.

[Tob01] Stephan Tobies. "Complexity results and practical algorithms for logics in knowledge representation". PhD thesis. RWTH Aachen University, Germany, 2001.

[Tra+14] Thanh-Luong Tran, Quang-Thuy Ha, Thi-Lan-Giao Hoang, Linh Anh Nguyen, and Hung Son Nguyen. "Bisimulation-Based Concept Learning in Description Logics". In: *Fundam. Inform.* 133.2-3 (2014), pp. 287–303.

[Tri+15] Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. "Optimising resolution-based rewriting algorithms for OWL ontologies". In: *J. Web Sem.* 33 (2015), pp. 30–49.

[Var98] Moshe Y. Vardi. "Reasoning about The Past with Two-Way Automata". In: *Proc. of ICALP.* Vol. 1443. LNCS. Springer, 1998, pp. 628–641.

[Zhu17] Dmitriy Zhuk. "A Proof of CSP Dichotomy Conjecture". In: *Proc. of FOCS.* 2017, pp. 331–342.

[ZKG18] Michael Zakharyaschev, Stanislav Kikot, and Olga Gerasimova. "Towards a Data Complexity Classification of Ontology-Mediated Queries with Covering". In: *Proc. of DL.* Vol. 2211. CEUR Workshop Proceedings. CEUR-WS.org, 2018.

[Zlo75] Moshé M. Zloof. "Query-by-Example: the Invocation and Definition of Tables and Forms". In: *Proc. of VLDB.* 1975, pp. 1–24.