Learning Queries under Description Logic Ontologies

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)
im Fachgebiet
Informatik

Vorgelegt

von M. Sc. Maurice Funk

geboren am 10.12.1995 in Bremen

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Carsten Lutz, Universität Leipzig
2. Prof. Dr. Diego Calvanese, FU Bozen, Italien

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 04.02.2025 mit dem Gesamtprädikat summa cum laude.

Contact address: mfunk@posteo.de

Abstract

This thesis investigates the learnability of conjunctive queries (CQs) under description logic (DL) ontologies. We focus on learning in the sense of Angluin's exact learning and of probably approximately correct (PAC) learning. In both models, the learner tries to learn a target query, of which only limited information is made available. In exact learning, this information is made available by a teacher who answers certain types of questions truthfully, whereas in PAC learning it is made available through randomly drawn labeled data examples.

We are then interested in algorithms that the learner can execute to always learn the target query in polynomial time, even when the information about the target query is provided with regard to a DL ontology. Our aim is to determine for which classes of conjunctive queries and for which ontology languages such polynomial time learning algorithms exist, and which kinds of questions (membership queries and equivalence queries) are necessary for polynomial time learning in the exact learning model. For this, we build upon existing results on the exact learnability of queries without ontologies.

We show that membership queries alone suffice to learn unary acyclic connected CQs (that correspond to $\mathcal{ELI}$ concepts) in polynomial time under *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ ontologies, if the interaction of functionality constraints and existential restrictions in the ontology is limited. In contrast, it turns out that an exponential number of membership queries is required to learn the target query reliably under many extensions of *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$, including those that permit conjunctions in concepts like $\mathcal{EL}^r$.

Furthermore, we consider teachers that answer both membership queries and equivalence queries, and show that chordal and symmetry-free CQs (and relevant subclasses thereof) are polynomial time learnable under $\mathcal{EL}^r$ ontologies in this setting. This result does not extend to $\mathcal{ELI}$ ontologies, under which it turns out that already simple query classes are not polynomial time learnable.

Finally, we review results that equivalence queries alone are not sufficient to learn simple path-shaped queries in polynomial time, unless NP = RP. Instead, we show that sample-efficient PAC learning of queries under ontologies is possible using a bounded fitting approach. We implement such an algorithm for tree-shaped CQs (that correspond to $\mathcal{EL}$ concepts) under $\mathcal{ELH}^r$ ontologies and show that the implementation compares favorably to an existing $\mathcal{EL}$ concept learning algorithm.

Acknowledgments

This dissertation is the result of years of work and would not have been possible without the support and encouragement of many great people, both during my time as a PhD student and before. I would like to express my heartfelt gratitude for this here.

First, I would like to thank those who have guided my academic journey these last years: My PhD advisor, Carsten Lutz, who has consistently directed me to interesting questions and who has shown me only encouragement, patience and support. Jean Jung, who advised my master's thesis (which continues to serve as a foundation for my work) and introduced me to research in the area of description logics. Thomas Schneider, whose classes were the highlight of my undergraduate years and who introduced me to various areas of theoretical computer science, including description logic.

A big thanks goes to my colleagues in Bremen and Leipzig for making the office at the university a nice and fruitful place to work in. Thank you, Gustav, Jean, Leif, Lukas, Marcin, Moritz, Quentin, Simon, Thomas, for many interesting questions and answers, discussions, shared coffees, and fun board games. I especially want to thank Simon for reading large parts of my dissertation and many helpful remarks.

My family I thank for always supporting me throughout my studies, and dealing with my unfortunate inability to explain my research and the content of this dissertation.

I am also very grateful to my friends, not limited to, but especially those in *Nerdkram* and *CDD*, for many fun adventures and memories. A special thanks goes to Yale for reading a draft of Chapter 1.

I would also like to thank Diego Calvanese for agreeing to review my dissertation.

Lastly, I would like to thank Alina, without whom no step of this journey would have been possible.

Contents

Chapter 1

Introduction

Relational databases are one of the major success stories of computer science, allowing data to be represented in a structured and processable way through relations. We might represent the data "john and jane are dogs" and "john is cute" in a database as the facts

$$\mathsf{Dog}(\mathsf{john}),\ \mathsf{Dog}(\mathsf{jane}),\ \mathsf{Cute}(\mathsf{john})$$

where we use Dog and Cute as unary relation symbols. We can then query such a database using logical expressions. If we want to retrieve all dogs that are cute from the data, we might pose the *query*

$$q(x) \leftarrow \mathsf{Dog}(x) \wedge \mathsf{Cute}(x)$$

to our database and would receive as answer only john but not jane. Many popular languages for writing queries to databases, like SQL, correspond to logical expressions in some way. The above query could be written equivalently in SQL as:

```
SELECT id FROM Dog JOIN Cute USING (id);.
```

How can these expressions used to query databases be obtained? The obvious answer is that they are written manually by people who understand both the logical semantics of the query language and the domain of the data. However, not all users of databases are familiar with the data and not all users have the required logical expertise to formulate correct queries on their own. This situation was recognized already early in the history of relational databases, and means of providing automated support for query writing were developed [Zlo75a; Zlo75b].

Automated support becomes even more important when queries are not posed to databases alone, but to *knowledge bases*, as querying knowledge bases requires more logical expertise. In a knowledge base, data is combined with a so-called *ontology* that contains background knowledge in the form of logical statements. For example, if we know that in our domain every dog is cute, we could state this formally as

$$\mathsf{Dog} \sqsubseteq \mathsf{Cute}$$

and add this logical statement to our knowledge base.

This can serve multiple purposes. Commonly, background knowledge is used to obtain more complete answers to a query from incomplete data. If we retrieve answers from our knowledge base with the same query $q(x) \leftarrow \mathsf{Dog}(x) \wedge \mathsf{Cute}(x)$ as before, we now receive both john and jane as answers, although direct information about the cuteness of jane is missing from the data. Furthermore, ontologies are used for data integration, where they provide a uniform vocabulary to query data from multiple, heterogeneous data sources [Pog+08; Xia+18]. They also form a central component of the Semantic Web [HKR10].

The knowledge in ontologies is usually more complicated than stating that every dog is cute. Large knowledge bases like YAGO 4.5 contain many thousand such statements in their ontology, involving more complicated expressions [Suc+23]. This means that effort and logical expertise are required when working with ontologies or when formulating queries, which makes these tasks cost and time intensive. Hence, automated methods that support users in writing queries under ontologies are desirable. One way to provide this support is by *learning* queries.

Learning is, of course, an ever relevant topic in AI research, with many theoretical and practical results within the last decades. Query learning, as considered in this thesis, is a *supervised* task, as it builds upon input-output examples provided by a user. As learned classifiers, logical expressions such as queries have certain advantages over other forms of classifiers not based on logical formalisms: logical expressions are easier to inspect, to explain and to verify. These advantages also allow us to show strong formal guarantees for learning algorithms, using formal models of learning from *computational learning theory*.

In this thesis, we investigate the existence of efficient algorithms that learn queries under ontologies written in description logics.

Description Logic Ontologies

Description logics (DLs) are a family of knowledge representation languages originating in the 1980s from systems such as KL-ONE [BS85]. Most DLs can be viewed as decidable fragments of first order logic and are closely related to modal logics.

DL ontologies contain knowledge about *concepts* and *roles*. Concepts describe properties of things and are built from atomic symbols like Dog and Cute and constructors like $\sqcap$, which describes *and*. For example, $\mathsf{Dog} \sqcap \mathsf{Cute}$ is a concept that describes all things that are both a dog and cute. Roles describe relations between things. A role like isFriendOf could express that one thing is the friend of another thing. We can use roles to construct more complicated concepts like the concept $\exists \mathsf{isFriendOf}.\mathsf{Dog}$ which describes all things that are a friend of something that is a dog. We can write this concept equivalently as the first order formula

$\varphi(x) = \exists y.\mathsf{isFriendOf}(x, y) \wedge \mathsf{Dog}(y)$ using Dog as a unary predicate and isFriendOf as a binary predicate.

DL ontologies are then sets of statements about these concepts and roles. The most common form of statement is *concept inclusion*, which allows us to express "every A is a B" knowledge. The concept inclusion $\mathsf{Dog} \sqsubseteq \mathsf{Cute}$ indicates that every dog is cute. Using the connection to first order logic, this inclusion can be equivalently written as the sentence

$$\forall x.\Big(\mathsf{Dog}(x) \rightarrow \mathsf{Cute}(x)\Big).$$

The various DLs differ in the ways complex concepts can be constructed from simpler ones and in the kinds of statements about concepts that can be made, and thus in the kinds of knowledge they can express. For example, some DLs, like *DL-Lite*$_{\text{core}}$, do not allow concepts to be combined with $\sqcap$, while $\mathcal{EL}$ does. Each DL gives rise to a separate *ontology language*, with its own expressiveness and computational properties.

One defining feature of knowledge bases that contain ontologies is the *open world assumption*, meaning that in contrast to traditional databases (which employ the *closed world assumption*), the absence of a fact in the data does not necessarily mean that its negation is true. For example, say that our data contains the fact Dog(jane), meaning that jane is a dog, but does not contain the fact Cute(jane), meaning that jane is cute. We then cannot conclude that Cute(jane) is false, but must consider that we do not know whether jane is cute. In fact, concluding that Cute(jane) is false in this case would contradict the background knowledge in our ontology that all dogs are cute. To enunciate this difference to the closed world assumption, we refer to our data as *ABoxes* (assertional boxes), that contain assertions about the world, as is usual in the area of DL.

As DL concepts are themselves logical expressions, we can use them as queries for knowledge bases, as so-called *instance queries*. If we query data for instances of the concept $\mathsf{Dog} \sqcap \mathsf{Cute}$, we get the same answers as for the query $q(x) \leftarrow \mathsf{Dog}(x) \wedge \mathsf{Cute}(x)$. Hence, algorithms that are able to learn certain kinds of queries efficiently can also be used to learn concepts. *Concept learning* is itself an active field of research, for much the same reasons as query learning: creating and extending ontologies are difficult and costly tasks that require both logical expertise and domain knowledge. Systems like DL-Learner [BLW16] and Ontolearn [DN23] aim to construct concepts from examples.

In this thesis, we focus on query learning under ontologies written in DLs from the $\mathcal{EL}$ and *DL-Lite* families. Both are limited in their expressivity, but are popular due to their favorable computational properties, making them suitable for reasoning about large ontologies and for use with large amounts of data.

DL ontologies of the $\mathcal{EL}$ and *DL-Lite* families are used for various purposes. For example, they form the logical basis of certain profiles of the OWL 2 web ontology language[1]. $\mathcal{EL}$ is the core of the *OWL 2 EL* profile and allows efficient reasoning over the knowledge in an ontology. Large biomedical ontologies like SNOMED CT[2] and GALEN are formulated in (dialects of) $\mathcal{EL}$ [RH97; Sch+09; SCC97].

DL-Lite forms the core of the *OWL 2 QL* profile and is designed to allow for efficient querying of data under ontologies [Cal+07]. DLs of the *DL-Lite* family are used in data intensive scenarios like data integration [Pog+08; Xia+18].

A formal definition of the description logics we use in this thesis and their semantics is given later in Chapter 3. For a thorough introduction to description logics, we refer to the textbooks [Baa+17] and [Baa+03].

Learning Queries

Learning a query can have various meanings. When we say that an algorithm learns queries, we must specify a *learning model* that defines what inputs the algorithm receives and what requirements we have for its outputs.

First, we have to specify a desired *class of queries* that the algorithm should produce. Depending on the use-case of query learning, some query classes can have more advantages than others, regarding understandability, interpretability, and learnability. The query $q(x) \leftarrow \mathsf{Dog}(x) \wedge \mathsf{Cute}(x)$ we have considered earlier belongs to the class of *conjunctive queries* (CQs). CQs are an extensively studied query class in the context of querying data under ontologies [BO15; Cal+13; CGL98; Eit+08; Gli+08]. They are a fragment of function-free first order logic with only conjunction and existential quantification. Equivalently, CQs correspond to the `SELECT-FROM-WHERE` fragment of SQL, and are the central element of the *basic graph patterns* of SPARQL. As such, many queries posed to relational databases are CQs, which makes CQs learning relevant for many scenarios [BO15; Cal+13]. We are also interested in subclasses of CQs, like the class of $\mathcal{EL}$ instance queries.

Then, in all learning models that we will introduce, information about the behavior of the query to be learned will be communicated through *data examples*. A data example like

$$\Big(\big\{\mathsf{Dog}(\mathsf{john}),\ \mathsf{Cat}(\mathsf{josie})\big\},\ \ \mathsf{john}\Big)$$

includes some data $\{\mathsf{Dog}(\mathsf{john}), \mathsf{Cat}(\mathsf{josie})\}$ and a potential answer john. For the query $q(x) \leftarrow \mathsf{Dog}(x)$, this is a *positive example*, since $\mathsf{Dog}(\mathsf{john})$ is included in the data. Otherwise, it would be a *negative example* for this query. Note that the examples do not refer to some fixed background database, but each comes with their own data.

[1] https://www.w3.org/TR/owl2-profiles/
[2] http://www.ihtsdo.org/snomed-ct

This allows us to specify the behavior of queries in all databases, and thus also to learn queries that yield the desired answers in all databases.

Specifically relevant for this thesis are the learning models of *fitting*, Angluin's *exact learning*, which will be our main focus, and Valiant's *probably approximately correct* (PAC) learning. Although these models seem quite different at first glance, we will see that they are closely connected.

To illustrate these learning models, and the effect of ontologies on learning, assume that we work under an ontology which includes the following statements:

$$\mathsf{Dog} \sqsubseteq \mathsf{Mammal}, \qquad \mathsf{Cat} \sqsubseteq \mathsf{Mammal},$$
$$\mathsf{Mammal} \sqsubseteq \mathsf{Animal}, \qquad \mathsf{Fish} \sqsubseteq \mathsf{Animal}.$$

These concept inclusions express that both dogs and cats are mammals, mammals are animals, and fish are animals.

Fitting In the fitting model, a learning algorithm receives data examples as input, which are labeled either positively or negatively. A *fitting algorithm* then needs to find a query such that all positively labeled data examples are positive examples for this query, and all negatively labeled data examples are negative examples for this query.

If a fitting algorithm receives the example $\big(\{\mathsf{Dog}(\mathsf{john})\}, \mathsf{john}\big)$ with a positive label, and the example $\big(\{\mathsf{Fish}(\mathsf{julia})\}, \mathsf{julia}\big)$, with a negative label, then it could return the query $q_H(x) \leftarrow \mathsf{Mammal}(x)$[3]. It could not return the query $q_H(x) \leftarrow \mathsf{Animal}(x)$, since this query yields the answer julia in the second example, which is labeled negatively.

This is a simple model of learning, that does not have strong requirements on the output query. Indeed, usually there are many queries that fit given data examples, and if the input data examples are labeled according to some query, the resulting fitting query need not be similar to that query. Especially, the fitting query could give answers that differ from the labeling query on all data that did not occur in data examples. The computation of fitting queries and concepts under ontologies is already well explored. In many cases, it has high computational complexity [Jun+22]. One way to deal with the high complexity is to give a learning algorithm access to more information, as in exact learning.

Exact Learning The exact learning model was first introduced by Angluin for learning regular languages from words [Ang87]. It is a model of *active learning*, in

[3]The H in q_H stands for hypothesis.

which two parties, the *learner* and the *teacher*, interact in a game-like fashion. The teacher has some target query in mind, and the learner aims to identify this target query by asking certain kinds of questions, which the teacher must answer truthfully. In our case, both parties are aware of the ontology. The hope is that a smart learner can ask the right questions and thus identify the target query quickly, more quickly than in a passive learning setting.

Assume that the teacher has the query $q_T(x) \leftarrow \mathsf{Animal}(x)$[4] in mind. The learner tries to identify this query by first asking

"Is julia an answer to q_T in the data {Fish(julia)}?".

In this case, the teacher responds with "Yes". Then, the learner could continue by asking

"Is q_T the query $q_H(x) \leftarrow \mathsf{Fish}(x)$?".

The teacher responds with "No" and gives an example where the two queries differ:

"q_T and q_H give different answers on the data {Dog(john)}"

The learner then concludes that john and jane must both be answers to q_T and tries again:

"Is q_T the query $q_H(x) \leftarrow \mathsf{Animal}(x)$?",

to which the teacher replies "Yes" — the learner has identified q_T.

Exact learning naturally models the situation where logical expertise and domain knowledge are not in the same hand, and a logic expert (the learner) constructs a query by interviewing a domain expert (the teacher). In the above example, two kinds of questions are used: the first question is a so-called *membership query*, and the second and third questions are so-called *equivalence queries*[5]. These two are the most common kinds of questions that are considered in exact learning. We formally define them later.

Of course, the learner and the teacher do not need to be human in the exact learning model. In fact, we are looking for learning algorithms that play the role of the learner and always identify the target query in little time. Furthermore, exact learning has been successfully applied in settings where, for example, the teacher is not a human, but a trained neural network [Blu+23]. Exact learning of queries has already been investigated for multiple query classes [tCDK13; tCD22], but not yet under ontologies.

[4] The T in q_T stands for target.

[5] Some literature on learning queries calls these membership oracle and equivalence oracle to avoid confusion with the queries that are learned.

PAC Learning The PAC model was introduced by Valiant [Val84] in the context of learning Boolean functions. In this model, the learner receives data examples that are drawn independently from some probability distribution and labeled according to a target query. Then, the learner needs to, with high probability, find a query that gives approximately the same answers as the target query on new examples drawn from the same distribution.

Assume that the target query is $q_T(x) \leftarrow \mathsf{Mammal}(x)$ and that we observe the dog john with probability 0.9 and the cat joan with probability 0.1. Equivalently, the positive data example $\big(\{\mathsf{Dog(john)}\},\ \mathsf{john}\big)$ is drawn with probability 0.9 from the example distribution and the positive example $\big(\{\mathsf{Cat(joan)}\},\ \mathsf{joan}\big)$ is drawn with probability 0.1. Then, if we draw a small sample from the distribution, it is likely that the sample exclusively contains dogs. Since the learner only receives dogs as examples and all dogs are labeled positively, it might output the query $q_H(x) \leftarrow \mathsf{Dog}(x)$, which is not equivalent to q_T, but differs from q_T on only 10 % of the examples drawn from the distribution.

In contrast to exact learning, a PAC learning algorithm does not require a teacher who is able to answer questions, but also does not exactly identify the target query, only a query that is with high probability close enough. Different from fitting algorithms, a PAC learning algorithm is required to *generalize* from the input examples, to other examples from the same distribution. In general, how close the result of a PAC learning algorithm is to the target query depends on the size of the sample it receives.

For all these learning models, there are often naive learning algorithms that try all possibilities until they succeed. These kinds of learning algorithms are not useful in practice, and hence we demand a notion of efficiency from our learning algorithms. As usual in computer science, we consider algorithms with running times that grow polynomially with the size of the input to be efficient or tractable, and algorithms with running times that grow exponentially with the size of the input to be inefficient or intractable. Later, we define the introduced models of learning formally and discuss how they are related.

Overview

The results of this thesis mainly focus on the exact learning model. Due to the established connections between the models, we can draw from existing results concerning fitting of queries under ontologies, and our results have implications on the PAC learnability of queries under ontologies. Since the query classes we consider contain concept instance queries, our results also have implications on the learnability of concepts under ontologies.

Each choice of query class and ontology language gives rise to a different exact learning setting that poses different requirements to learning algorithms. One query class might be efficiently learnable under one ontology language, but that must not also be the case under another ontology language. And for a second query class, the situation might be reversed. Hence, to understand the possible choices in learning queries under ontologies, we aim to answer:

Which query classes are efficiently learnable under which ontology languages?

Additionally, it is important to understand which kinds of questions the teacher needs to be able to answer for a query class to be efficiently learnable under an ontology language. We have seen both equivalence queries and membership queries. In practical scenarios, a teacher might only be able to answer certain kinds of questions, or a teacher who can answer all question could be too expensive to obtain. Hence, we also aim to answer:

Which questions needs the teacher to be able to answer for efficient learning of a query class under an ontology language?

The remainder of this thesis is structured as follows:

Chapter 2 We begin with an overview of related approaches and results in the areas of query, concept, and ontology learning.

Chapter 3 Then, we formally define the relevant query classes and ontology languages, and precisely state the learning models and relevant known results, as well as define what we mean by efficient learning.

Chapter 4 We begin the investigation into exact learning by considering only membership queries. We give efficient learning algorithms for several combinations of query class and ontology language in this setting, and show that certain other combinations are not efficiently learnable.

Chapter 5 Then, we investigate exact learning with both membership queries and equivalence queries. This allows us to formulate efficient learning algorithms for query classes and ontology languages that were not efficiently learnable using only membership queries.

Chapter 6 Finally, we consider PAC learning of queries, which has a strong connection to exact learning with only equivalence queries. We show that many query classes are not efficiently PAC learnable using the usual notion of efficiency based on running time, but still PAC learnable from a small number of data examples. We also describe SPELL, our implementation of a PAC learning algorithm.

Chapter 7 We conclude with a summary of the results and comment on future directions.

We obtain the following results, concerning description logics of the *DL-Lite* and $\mathcal{EL}$ families as ontology languages and subclasses of CQs.

In Chapter 4 we show that ELIQs (unary, acyclic, and rooted CQs) are polynomial time exact learnable under $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontologies using only membership queries (Theorem 4.42), where the $\mathcal{F}-$ indicates a restriction in the interaction between functionality constraints and existential restrictions. Furthermore, we show that ELIQs are not polynomial query learnable if this restriction is lifted (Theorem 4.29), or if the ontology language can use conjunctions (Theorem 4.28). As an intermediate step, we also show the independently interesting results that ELIQs permit construction of frontiers under $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontologies in polynomial time (Theorem 4.23), and that certain generalizing sequences of CQs are bounded in length under ontologies (Theorem 4.35).

In Chapter 5 we show that conjunctions in the ontology language can be handled with equivalence queries. We first show that ELIQs are polynomial time exact learnable under $\textit{DL-Lite}_{\text{horn}}^{\mathcal{F}^-}$ ontologies (Theorem 5.17), and then that the large class of chordal, symmetry-free CQs (that contains non-unary CQs, cyclic CQs and CQs that are not rooted) is polynomial time exact learnable under $\mathcal{EL}^r$ ontologies (Theorem 5.48). We complement this by showing that no subclass of CQs that contains all ELQs is polynomial query learnable under $\mathcal{ELI}$ ontologies (Theorem 5.50), and that learning $\mathcal{ELU}$ instance queries is as hard as learning Boolean formulas (Theorem 5.52).

In Chapter 6 we review the result that no class of CQs that contains all path queries is polynomial time PAC learnable, unless RP = NP (Theorem 6.7). In contrast, we show that all combinations of ontology language and query class are PAC learnable with polynomial sample complexity (Lemma 6.9), using a bounded fitting approach. Certain other fitting algorithms, that aim for different properties than the smallest size, are not sample-efficient PAC learning algorithms (Theorems 6.18, 6.19 and 6.23). We describe an implementation of a bounded fitting based sample-efficient PAC learning algorithm (Theorem 6.10) for learning ELQs under $\mathcal{ELH}^r$ ontologies based on a SAT solver and present benchmark results that show that this implementation compares favorably with an existing approach for ELQ learning.

Related Publications

Parts of this thesis have already been published in the following workshop and conference papers. In the introductions of the respective chapters, we describe in more detail the relationship of the presented material to these publications.

[tCat+23a] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. *Extremal Fitting CQs Do Not Generalize*. Version 1. 2023. DOI: 10.48550/arXiv.2312.03407. arXiv: 2312.03407 [cs]. Pre-published.

[tCat+23b] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "SAT-based PAC Learning of Description Logic Concepts." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI 2023. Edited by Edith Elkind. International Joint Conferences on Artificial Intelligence, 2023, pages 3347–3355. DOI: 10.24963/ijcai.2023/373.

[tCat+24] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "On the Non-Efficient PAC Learnability of Conjunctive Queries." In: *Information Processing Letters* 183.106431 (2024). DOI: 10.1016/J.IPL.2023.106431.

[FJL21a] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Actively Learning Concepts and Conjunctive Queries under ELr-ontologies." In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. IJCAI 2021. Edited by Zhi-Hua Zhou. International Joint Conferences on Artificial Intelligence, 2021, pages 1887–1893. DOI: 10.24963/ijcai.2021/260.

[FJL21b] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Actively Learning ELI Queries under DL-Lite Ontologies." In: *Proceedings of the 34th International Workshop on Description Logics*. DL 2021. Edited by Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt. Volume 2954. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2954/paper-14.pdf (visited on 05/14/2024).

[FJL22a] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Exact Learning of ELI Queries in the Presence of DL-Lite-Horn Ontologies." In: *Proceedings of the 35th International Workshop on Description Logics*. DL 2022. Edited by Ofer Arieli, Martin Homola, Jean Christoph Jung, and Marie-Laure Mugnier. Volume 3263. CEUR Workshop Proceedings. CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3263/paper-9.pdf (visited on 05/14/2024).

[FJL22b] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Frontiers and Exact Learning of ELI Queries under DL-Lite Ontologies." In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. IJCAI-ECAI-2022. Edited by Luc De Raedt. International

Joint Conferences on Artificial Intelligence, 2022, pages 2627–2633.
DOI: 10.24963/ijcai.2022/364.

Chapter 2

Related Work

The learning of logical expressions such as queries and concepts has received attention in various settings. In this chapter, we review the approaches that are most relevant to this thesis and remark on connections. We begin in Section 2.1 with looking at approaches to fit concepts and queries to data, some of which take ontologies into account. Then, in Section 2.2 we review existing results on the exact learnability and PAC learnability of concepts and queries. Finally, we consider the connection to learning ontologies in Section 2.3.

2.1 Fitting Examples with Queries

The construction of queries that fit given data examples, such as in the *query-by-example* paradigm or in *query reverse engineering*, is an active topic of database research, both from a practical and a theoretical perspective [Mar19]. The problem has been investigated, for example, for CQs [BR17; tCD15], SPARQL queries over RDF data [ADK16], path queries over graph databases [BCL15], and tree-patterns [CW16]. As these investigations focus on databases, they do not take ontologies into account.

In the field of description logic, the construction of concepts that fit examples is also often desired for ontology engineering. One of the first and most developed approaches to this task is based on computing *most specific concepts* (MSCs) of the examples and computing the *least common subsumer* (LCS) of the obtained concepts [BKM99; CBH92]. However, under the usual semantics, MSCs of examples only exist for acyclic examples, and can otherwise only be approximated [KM02]. Additionally, the LCS of $\mathcal{EL}$ concepts under $\mathcal{EL}$ ontologies does not always exist when the ontology is cyclic [Baa03; BST07]. Zarrieß and Turhan characterize when $\mathcal{EL}$ the MSCs and LCSs exist under $\mathcal{EL}$ ontologies [ZT13]. Jung, Lutz, and Wolter determine the complexity of deciding whether the MSC or the LCS exists, and of verifying if a given concept is the MSC or the LCS, for both $\mathcal{EL}$ and $\mathcal{ELI}$ concepts, and under ontologies [JLW20].

Closest to the learning problems considered in this thesis among the exist-

ing research are the investigations into the fitting problem for queries under ontologies. This has been investigated for various query classes and ontology languages [Fun+19; GJS18; Jun+20; Jun+22; Ort19], See also the PhD thesis of Pulcini [Pul22] for an overview of the results on fitting queries under ontologies. We will review the relevant results later in Section 3.2.

In many cases, no query exists that fits data examples exactly, and therefore approximations of the notion of fitting have also been considered by Cima, Croce, and Lenzerini [CCL21].

Refinement-based Search and Inductive Logic Programming

In practice, systems like DL-Learner or DLFoil, that computing fitting description logic concepts or similar queries from labeled data examples, with or without an ontology, are often based on ideas from *inductive logic programming* (ILP), like refinement operators and the FOIL algorithm.

In general, ILP is concerned with learning logic programs that entail a given set of positive example facts but none of the given negative example facts under a set of background facts[1] [ND97]. In the context of learning queries as used in this thesis, we can think of the background facts as an ABox, and of the logic program as the query that is to be learned. Then, the resulting query should return all the positive facts as answers and none of the negative facts.

In this sense, the ILP literature has obtained fitting algorithms as well as positive and negative PAC learnability results for various classes of CQs that are defined, for example, by limitations on the use of existential variables, determinacy conditions and restricted variable depth. An overview can be found in [ND97, Chapter 18]. These query classes are orthogonal to the classes of ELQs and ELIQs relevant in our setting.

ILP algorithms are often based on specific *refinement operators*. A (downward) refinement operator ρ takes as input a query q and returns a set of *specializations* of q. Symmetrically, an upward refinement operator returns *generalizations* of q. Refinement operators are then the basis of a search procedure that, starting from some initial query, aims to find a query that fits all examples, by specializing to exclude all negative examples, or by generalizing to include all positive examples. In order for this search procedure to eventually arrive at a fitting query, it is often required that the used refinement operator is *finite*, *proper*, and *complete*.

Badea and Nienhuys-Cheng first proposed to use refinement operators to find fitting description logic concepts. They show that there is a complete but not finite refinement operator for concepts of the description logic $\mathcal{ALER}$ and argue that no

[1] At least in the so-called *normal problem setting*

complete and finite refinement operator for $\mathcal{ALER}$ can exist [BN00]. This is also the case for concepts of other expressive description logics like $\mathcal{ALC}$ where finite, proper, and complete refinement operators also do not exist, already in the case without ontologies [LH10].

For $\mathcal{EL}$ concepts, there exist refinement operators that are finite, proper, and complete [LH09], but it is known that no such operator exists for $\mathcal{EL}$ concepts under $\mathcal{EL}$ ontologies [Kri19]. Only if the considered $\mathcal{EL}$ ontologies are of a restricted form, $\mathcal{EL}$ concept refinement operators can be finite, proper, and complete [LH09].

Important in this area is also the work of Kriegel [Kri18a; Kri18b; Kri21] that deals with the structure of the $\mathcal{EL}$ *subsumption lattice* that is implicitly traversed by search procedures to find a fitting $\mathcal{EL}$ concept. In this structure, concepts have direct upwards (downwards) neighbor concepts that are minimally more general (more specific). Even under the empty ontology, there are $\mathcal{EL}$ concepts of size n that have a number of downward neighbors that is exponential in n, and there are $\mathcal{EL}$ concepts of size linear in n such that they can only be reached from $\top$ by a number of neighborhood-steps that is n-fold exponential in n. Hence, a complete search procedure for fitting $\mathcal{EL}$ concepts with refinement operators can be infeasible in certain situations.

Nonetheless, implementations of search procedures using refinement operators together with heuristics are in many cases able to quickly find fitting concepts or approximately fitting concepts. These algorithms aim for various degrees of completeness, depending on their use case. Such systems and algorithms are for example DL-Learner [BLW16], DL-FOIL [FdAE08; Fan+18], YINYANG [IPF07], DL-FOCL [RFdA20], and DRILL [DN23].

Other approaches for learning DL concepts not based on refinement operators are using answer set programming [Lis12; Lis16], learning $\mathcal{ALC}$ concepts using bisimulations [Tra+14], and trade off accuracy of the fitting concept for efficiency [SH19]. Except for the basic setting of fitting concepts, these do not have a strong relation to the results in this thesis.

2.2 Exact Learning of Queries

Closest to the subject of this thesis are existing results on exact learnability and PAC learnability of queries or concepts in the setting without ontologies. We introduce the most important ones briefly.

Ten Cate, Dalmau, and Kolaitis show that *Global-As-View schema mappings* (GAV schema mappings) are polynomial time exact learnable using membership queries and equivalence queries. A GAV schema mapping is a set of logical expressions of

the form

$$\forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})),$$

where φ is a conjunction of atoms and ψ is a single k-ary atom. If in all expressions of a GAV schema mapping the atom ψ is identical, we can view it as a union of CQs (UCQ). Hence, UCQs are also polynomial time exact learnable using membership queries and equivalence queries [tCDK13]. Additionally, the learning algorithm of ten Cate, Dalmau, and Kolaitis can be modified to obtain the same result for CQs. The UCQ learning algorithm bears resemblance to the foundational exact learning algorithm for propositional Horn formulas by Angluin, Frazier, and Pitt [AFP92], which also inspired, for example, exact learning algorithms for first-order Horn formulas [AK02].

Furthermore, ten Cate, Dalmau, and Kolaitis show that CQs and UCQs are only polynomial time exact learnable using both membership queries and equivalence queries, only one type of query does not suffice [tCDK13; tCat+18]. Indeed, only subclasses of CQs, like the class of all ELIQs, fulfill the important precondition of being *uniquely characterizable* by data examples and are therefore polynomial time learnable using only membership queries [tCD22]. In Chapters 4 and 5, we build on these results and extend them to cases with ontologies. We review them there in more detail.

Fortin et al. investigate the unique characterizability of linear temporal logic (LTL) formulas. While many LTL formulas are not uniquely characterizable, they show that several fragments are. They also identify combinations of LTL and ELIQs that are uniquely characterizable, and show that these combinations are polynomial time learnable with only membership queries [For+22]. Jung et al. extend these results and consider the unique characterizability and learnability of queries that are combinations of LTL and ELIQs under DL ontologies [Jun+23a; Jun+23b]. Thereby, they also generalize the techniques and results presented in Chapter 4.

Ten Cate and Koudijs investigate the unique characterizability of fragments of modal logic with data examples. They show that the fragment of positive modal formulas that only use $\Diamond$, that are similar to ELQs, and the fragment of positive modal formulas that only use $\Box$ are uniquely characterizable, while their union or the full modal language are not finitely characterizable [tCK23; Kou22].

Haussler [Hau89] and Kietz [Kie93] show that acyclic CQs are not polynomial time PAC learnable from data examples unless RP = NP, and therefore also not exact learnable using only equivalence queries. We revisit their proofs in Chapter 6. Hirata shows that acyclic CQs that use ternary atoms are not polynomial time PAC predictable from data examples[2] under certain cryptographic assumptions [Hir00; Hir05]. PAC prediction is a related learning model to PAC learning, where the

[2]Data examples are called *extended instances* in the work of Hirata.

learning algorithm need not output a hypothesis from a specific query class, but can output any algorithm that classifies examples in polynomial time. PAC learnability implies PAC predictability for classes of queries that can be answered in polynomial time. PAC prediction is also an interesting learning model in itself with many interesting results, but not further considered in this thesis.

Learning CLASSIC Concepts

In the area of description logics, PAC learnability (with and without membership queries) of concepts was investigated for the early description logic CLASSIC, without considering any form of ontologies. CLASSIC concepts are built from many constructors not contained in $\mathcal{EL}$ or *DL-Lite* like universal restriction $\forall r.C$ and the *same-as* constructor that demands that two role paths end at the same individual [Bor+89]. However, CLASSIC has no way to express existential restrictions, which is the core feature of $\mathcal{EL}$, and hence the expressive power of CLASSIC is incomparable to the one of $\mathcal{EL}$.

Cohen and Hirsh consider PAC learnability of CLASSIC concepts from concept examples that are labeled according to subsumption of the target concept. Specifically, a concept D is a positive example of a target concept C_T if $D \sqsubseteq C_T$ and a negative example if $D \not\sqsubseteq C_T$. Assuming RP $\neq$ NP, they show that already CoreCLASSIC concepts, which are CLASSIC concepts that use only conjunction, universal quantification and *same-as*, are not polynomial time PAC learnable from concept examples [CH92; CH94b; CH95]. Consequently, CoreCLASSIC concepts are also not polynomial time learnable with only equivalence queries, where the counterexamples returned from equivalence queries are concepts. It is interesting to note that this lower bound uses a definition of PAC learning in which the running time and the sample size may depend on the size of the target concept and the size of the examples, similar to the definition we will use.

Cohen and Hirsh also show that the fragment of CLASSIC without *same-as* and role inclusions, called C-CLASSIC, is polynomial time PAC learnable from concept examples [CH94a]. The key element of their learning algorithm is the ability to compute the least common subsumer of C-CLASSIC concepts in polynomial time.

Frazier and Pitt consider exact learnability of CLASSIC concepts in Angluin's learning framework with membership queries and equivalence queries that both use concept examples. The fragment of CLASSIC they consider includes both the aforementioned CoreCLASSIC and C-CLASSIC. They show that CLASSIC concepts are not exact learnable in polynomial time with concept membership queries alone, but can be learned in polynomial time using both concept membership queries and equivalence queries [FP96]. Their learning algorithm works similarly to the ones we discuss in Chapter 5, constructing products to compute commonalities

with counterexamples and then minimizing the result using membership queries. However, it does not need to take ontologies into account. Since the expressive powers of $\mathcal{EL}$ and CLASSIC are incomparable, and the permitted membership queries and equivalence queries differ, results on learning CLASSIC cannot be transferred to learning $\mathcal{EL}$ concepts or CQs.

2.3 Learning Description Logic Ontologies

Related to the learning of concepts or queries under ontologies, is the learning of entire ontologies. This has been investigated in multiple settings that differ in the choice of examples. The aim of exact learning of ontologies is to identify unknown target ontologies in polynomial time through interaction with a teacher. To achieve polynomial time, lightweight ontology languages of the $\mathcal{EL}$ and *DL-Lite* families are considered as targets, since they allow for polynomial time reasoning.

In the first setting, the examples are *concept inclusions*. A concept inclusion $C \sqsubseteq D$ is a positive example for a target ontology $\mathcal{O}_T$ if $\mathcal{O}_T \models C \sqsubseteq D$ and a negative example if $\mathcal{O}_T \not\models C \sqsubseteq D$. Konev et al. investigate the exact learnability of ontologies using concept inclusion membership and equivalence queries. They show that $\mathcal{EL}$ ontologies are not learnable with a polynomial number of queries, but two fragments of $\mathcal{EL}$, $\mathcal{EL}_{\mathsf{rhs}}$ and $\mathcal{EL}_{\mathsf{lhs}}$, are learnable in polynomial time [Kon+18]. In $\mathcal{EL}_{\mathsf{lhs}}$ ontologies, the right-hand side of concept inclusions must be a concept name, and in $\mathcal{EL}_{\mathsf{rhs}}$ ontologies, the same restriction holds for the left-hand side.

In the next setting, the examples are *data retrieval examples*. A data retrieval example is a tuple $(\mathcal{A}, q, a)$ with $\mathcal{A}$ an ABox, q a query, and a an individual from $\mathcal{A}$. It is a positive example for a target ontology $\mathcal{O}_T$ if $\mathcal{A}, \mathcal{O}_T \models q(a)$ and a negative example if $\mathcal{A}, \mathcal{O}_T \not\models q(a)$. Konev, Ozaki, and Wolter show that in many cases, exact learning with data retrieval example membership queries and equivalence queries reduces to the case with concept inclusion examples, and thus obtain similar positive results: $\mathcal{EL}_{\mathsf{lhs}}$ and $\mathcal{EL}_{\mathsf{rhs}}$ ontologies are polynomial time learnable using ELQs or ELIQs in data retrieval examples, but $\mathcal{EL}$ ontologies are not learnable with a polynomial number of membership and equivalence queries using ELQs [KOW16].

Finally, another option is to learn ontologies with data retrieval examples, but where equivalence queries are limited to checking equivalence over a single fixed ABox. This does not necessarily result in an ontology that is equivalent to the target ontology, but to one that is *query inseparable* over this fixed ABox. In this setting, $\mathcal{EL}$ ontologies are polynomial time learnable using ELQs in the data retrieval examples [OPM20].

At first glance, learning ontologies seems to be closely related to learning concepts under ontologies, since an $\mathcal{EL}$ ontology may contain many $\mathcal{EL}$ concepts, and hence

any ontology learning algorithm has to construct concepts. However, there is no apparent natural reduction from any of these settings to the setting used in this thesis, and therefore no easy way to use the aforementioned positive and negative results. The challenge of such a reduction lies in the fact that ontology learning algorithms can produce arbitrarily structured ontologies as hypotheses, but the concept learning teacher requires a single concept as a hypothesis.

There are also approaches to learning ontologies that are not based on exact learning, and hence further removed from the learning setting considered in this thesis. Most prominent is the mining of concept inclusions from finite interpretations, inspired by *formal concept analysis*. There, a finite interpretation is taken as input and the aim is to compute an ontology that is a finite axiomatization of all concept inclusions that hold in this interpretation [BD08]. This procedure allows the extraction of ontological knowledge from existing data and, in contrast to exact learning, does not inherently require interaction with a teacher. Existing work mostly focuses on $\mathcal{EL}$ ontologies, and as $\mathcal{EL}$ axiomatizations may be of exponential size, many techniques have been developed to compute axiomatizations efficiently, or to approximate them [BD09; BDK16; Gui+21; Kri24].

The survey [Oza20] lists and compares these and other approaches to ontology learning.

Chapter 3

Preliminaries

Before we begin to look into the learnability of queries under description logic ontologies, we first need to review relevant concepts and define what we mean when we say that something is learnable. In Section 3.1 we define the syntax and semantics of the description logic ontology languages and query classes we use, and review some of their properties. Then, in Section 3.2 we formally define the fitting problem and review known results about its computational complexity. We continue in Section 3.3 with the definition of the exact learning model for our setting and discuss some of its properties. Finally, in Section 3.4 we define the PAC model of learning and formally connect it to the exact learning model.

3.1 Description Logic and Conjunctive Queries

To present results for ontologies written in DLs of the $\mathcal{EL}$ and *DL-Lite* families in a unified way, we define all relevant DLs as sublanguages of the DL $\mathcal{ELIHF}_\bot$.

Description Logic

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be countably infinite sets of *concept names*, *role names*, and *individual names*, respectively. We use A, B for concept names, r, s, t for role names and a, b for individual names, with additional subscripts when necessary. A *role* R takes the form r or r^- where r is a role name and r^- is called an *inverse role*. If $R = s^-$ is an inverse role, then R^- denotes the role name s.

An $\mathcal{ELI}$ *concept* is formed according to the syntax rule

$$C, D ::= \top \mid A \mid C \sqcap D \mid \exists R.C$$

where A ranges over $\mathsf{N_C}$ and R over roles. An $\mathcal{EL}$ *concept* is an $\mathcal{ELI}$ concept that does not use inverse roles. We refer to $\sqcap$ as *conjunction*, and $\exists$ as *existential restriction*. We call an existential restriction $\exists R.C$ *unqualified* if $C = \top$ and *qualified* if $C \neq \top$.

An $\mathcal{ELIHF}_\bot$ *ontology* $\mathcal{O}$ is a finite set of *concept inclusions* $C \sqsubseteq D$, *role inclusions* $R \sqsubseteq S$, (*global*) *functionality constraints* $\mathsf{func}(R)$, *role disjointness constraints* $R \sqcap S \sqsubseteq \bot$,

and *concept disjointness constraints* $C \sqcap D \sqsubseteq \bot$, where C and D are $\mathcal{ELI}$ concepts and R and S are roles. We use $C \equiv D$ as a shorthand for $C \sqsubseteq D$ and $D \sqsubseteq C$, and do the same for roles.

The semantics of $\mathcal{ELIHF}_\bot$ concepts and ontologies is defined in terms of interpretations. An interpretation $\mathcal{I}$ is a tuple $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is the domain, a non-empty set that may be finite or infinite, and $\cdot^\mathcal{I}$ is the interpretation function. The interpretation function assigns to each concept name $A \in \mathsf{N_C}$ a subset of $\Delta^\mathcal{I}$, to each role name $r \in \mathsf{N_R}$ a binary relation over $\Delta^\mathcal{I}$ and to each individual name $a \in \mathsf{N_I}$ an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. The interpretation function can be extended to assign each $\mathcal{ELI}$ concept a subset of $\Delta^\mathcal{I}$ by setting

$$\begin{aligned} r^{-\mathcal{I}} &= \{(e,d) \mid (d,e) \in r^\mathcal{I}\}, \\ \top^\mathcal{I} &= \Delta^\mathcal{I}, \\ (C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I}, \text{ and} \\ (\exists R.C)^\mathcal{I} &= \{d \mid (d,e) \in R^\mathcal{I} \text{ and } e \in C^\mathcal{I}\}. \end{aligned}$$

An interpretation $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$, a role inclusion $R \sqsubseteq S$ if $R^\mathcal{I} \subseteq S^\mathcal{I}$, a functionality constraint $\mathsf{func}(R)$ if $R^\mathcal{I}$ is a partial function, a role disjointness constraint $R \sqcap S \sqsubseteq \bot$ if $R^\mathcal{I} \cap S^\mathcal{I} = \varnothing$, and a concept disjointness constraint $C \sqcap D \sqsubseteq \bot$ if $C^\mathcal{I} \cap D^\mathcal{I} \sqsubseteq \bot$.

An interpretation $\mathcal{I}$ is a *model* of an $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}$, if it satisfies all inclusions and constraints in $\mathcal{O}$. For any concept inclusion, role inclusion, functionality constraint or role disjointness constraint α we write $\mathcal{O} \models \alpha$ if every model of $\mathcal{O}$ satisfies α. For two concepts C, D we write $\mathcal{O} \models C \equiv D$ if $\mathcal{O} \models C \sqsubseteq D$ and $\mathcal{O} \models D \sqsubseteq C$.

We additionally restrict the interaction between role inclusions and functionality constraints in $\mathcal{ELIHF}_\bot$ ontologies. If $\mathcal{O} \models R \sqsubseteq S$ for roles $R \neq S$, then $\mathsf{func}(S) \notin \mathcal{O}$. This restriction of the interaction of role inclusions and functionality constraints implies that $\mathcal{O} \models \mathsf{func}(R)$ if and only if $\mathsf{func}(R) \in \mathcal{O}$ for all roles R. Moreover, it corresponds to the restriction (A_3) used in [Art+09] for the DL *DL-Lite*$_{\text{core}}^{(\mathcal{HF})}$ in order to make satisfiability checking and the data-complexity of query answering tractable. The restriction is also adopted for *DL-Lite* in [Kon+10]. We adopt it for similar reasons, and note that this makes our version of $\mathcal{ELIHF}_\bot$ non-standard. We point out when results do not hold in absence of this restriction.

The inclusion of role disjointness constraints and concept disjointness constraints (instead of allowing $\bot$ as a concept constructor), as well as the above restriction of role inclusions and functionality constraints, allows us to present several sublanguages of $\mathcal{ELIHF}_\bot$ uniformly. We define these languages next.

- An $\mathcal{ELI}$ *ontology* is an $\mathcal{ELIHF}_\bot$ ontology that contains only concept inclusions.

- An $\mathcal{EL}^r$ *ontology* is an $\mathcal{ELI}$ ontology where inverse roles may only occur in concept inclusions of the form $\exists r^-.\top \sqsubseteq A$ where r is a role name and A is a concept name. The ^{r} stands for *range restrictions*, as concept inclusions of the form $\exists r^-.\top \sqsubseteq A$ restrict the range of roles. The corresponding *domain restrictions* can be achieved without use of an inverse role as $\exists r.\top \sqsubseteq A$. We will also mention $\mathcal{EL}$ *ontologies*, which do not permit range restrictions, and $\mathcal{ELH}^r$ *ontologies*, that are $\mathcal{EL}^r$ ontologies that may also use role inclusions.

- A *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ *ontology* is an $\mathcal{ELIHF}_\bot$ ontology where all occurring existential restrictions are unqualified and that does not contain any role inclusions. For ontology languages of the *DL-Lite*-family, unqualified existential restrictions $\exists R.\top$ are usually written as $\exists R$, omitting the $\top$, and disjointness constraints $A \sqcap B \sqsubseteq \bot$ are written as $A \sqsubseteq \neg B$. We use the former notation for uniformity with DLs of the $\mathcal{EL}$-family.

- A *DL-Lite*$_{\text{horn}}$ *ontology* is a *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ ontology that does not contain functionality constraints.

- A *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ *ontology* is an $\mathcal{ELIHF}_\bot$ ontology where all occurring existential restrictions are unqualified, and that does not contain any conjunctions. Note that due to the restriction of the interaction of role inclusions and functionality constraints, this language corresponds to *DL-Lite*$^{(\mathcal{HF})}_{\text{core}}$ of [Art+09]. The same restriction is employed in *DL-Lite*$_{\mathcal{A}}$ [Pog+08].

  Note that role inclusions allow *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ ontologies to express a limited form of qualified existential restrictions. The concept inclusion $A \sqsubseteq \exists r.B$ can be expressed in *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ through an additional role name r_B with $A \sqsubseteq \exists r_B.\top$, $r_B \sqsubseteq r$, and $\exists r_B^-.\top \sqsubseteq B$.

- A *DL-Lite*$_{\text{core}}$ *ontology* is a *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ ontology that does not contain any role inclusions and functionality constraints.

- A *conjunctive ontology* (conj) is an $\mathcal{ELI}$ ontology without existential restrictions, that is, all concept inclusions have the form

  $$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B_1 \sqcap \cdots \sqcap B_n$$

  with $A_1, \ldots, A_n, B_1, \ldots B_n$ concept names.

We refer to these languages as *ontology languages*. Their definitions are summarized in Table 3.1. Figure 3.1 shows an overview of the relationships of all these languages, where an arrow indicates that one language is syntactically contained in another.

Table 3.1: Overview of the features of the relevant ontology languages.

$\mathcal{L}$	r^-	$C \sqcap D$	$\exists R.C$	$\exists R.\top$	$\mathsf{func}(R)$	$R \sqsubseteq S$	$\sqsubseteq \bot$
$\mathcal{ELIHF}_\bot$	✓	✓	✓	✓	✓	✓	✓
$\mathcal{ELI}$	✓	✓	✓	✓			
$\mathcal{EL}^r$		✓	✓	✓			
$DL\text{-}Lite^{\mathcal{F}}_{\text{horn}}$	✓	✓		✓	✓		✓
$DL\text{-}Lite_{\text{horn}}$	✓	✓		✓			✓
$DL\text{-}Lite^{\mathcal{HF}}_{\text{core}}$	✓			✓	✓	✓	✓
$DL\text{-}Lite_{\text{core}}$	✓			✓			✓
conj		✓					

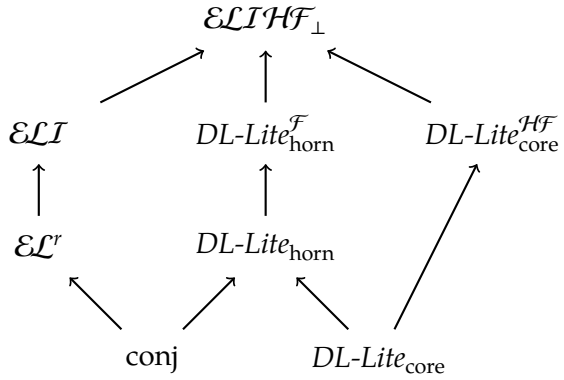

Figure 3.1: Relationship of the sublanguages of $\mathcal{ELIHF}_\bot$.

ABoxes

An *ABox* $\mathcal{A}$ is a finite set of concept assertions $A(a)$ and role assertions $r(a, b)$ with A a concept name, r a role name, and a, b individual names We write $\mathsf{ind}(\mathcal{A})$ for the set of all individual names that occur in an ABox $\mathcal{A}$. A *pointed ABox* is a tuple $(\mathcal{A}, \overline{a})$ of an ABox $\mathcal{A}$ and a tuple of individual names $\overline{a} \in \mathsf{ind}(\mathcal{A})^*$. In general, we always use the notation $\overline{\circ}$ to refer to a tuple. In the context of learning, we refer to a pointed ABox as a *(data) example*.

An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$, that is $a^\mathcal{I} \in A^\mathcal{I}$ if $A(a) \in \mathcal{A}$ and $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$ if $r(a, b) \in \mathcal{A}$. An ABox $\mathcal{A}$ is *satisfiable* under an ontology $\mathcal{O}$ if $\mathcal{A}$ and $\mathcal{O}$ have a common model. We make the *unique name assumption*, that is $a^\mathcal{I} \neq b^\mathcal{I}$ for all $a, b \in \mathsf{ind}(\mathcal{A})$ with $a \neq b$. This is relevant for satisfiability under functionality constraints: the ABox $\{r(a, b_1), r(a, b_2)\}$ is not

satisfiable under an ontology that contains the functionality constraint $\mathsf{func}(r)$, but can be if the unique name assumption is not adopted.

For a concept C, we write $\mathcal{A}, \mathcal{O} \models C(a)$ if for all models $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$. We can view a concept as a (*concept*) *instance query*, and say that a is an answer to the instance query C posed to the ABox $\mathcal{A}$ under the ontology $\mathcal{O}$ if $\mathcal{A}, \mathcal{O} \models C(a)$.

We can view an ABox $\mathcal{A}$ as a finite interpretation $\mathcal{I}_{\mathcal{A}}$ with $\Delta^{\mathcal{I}_{\mathcal{A}}} = \mathsf{ind}(\mathcal{A})$, $a \in A^{\mathcal{I}_{\mathcal{A}}}$ if $A(a) \in \mathcal{A}$, and $(a, b) \in r^{\mathcal{I}_{\mathcal{A}}}$ if $r(a, b) \in \mathcal{A}$, for all $r \in \mathsf{N_R}$ and $A \in \mathsf{N_C}$. Analogously, every finite interpretation $\mathcal{I}$ can be viewed as the ABox obtained by including the assertion $A(a)$ if $a \in A^{\mathcal{I}}$ and the assertion $r(a, b)$ if $(a, b) \in r^{\mathcal{I}}$.

The *underlying directed graph* $G_{\mathcal{A}}$ of an ABox $\mathcal{A}$ has the vertices $\mathsf{ind}(\mathcal{A})$ and the edges $\{(a, b) \mid r(a, b) \in \mathcal{A}\}$. We say that an ABox $\mathcal{A}$ is *acyclic* if $G_{\mathcal{A}}$ is acyclic and there are no multi-edges: $r(a, b) \in \mathcal{A}$ implies that $s(a, b) \notin \mathcal{A}$ for all role names s with $s \neq r$. We say that an ABox $\mathcal{A}$ is *connected* if $G_{\mathcal{A}}$ is connected. We say that a pointed ABox $(\mathcal{A}, \overline{a})$ is *rooted* if each connected component of $G_{\mathcal{A}}$ contains an element of $\overline{a}$. We say that $\mathcal{A}$ is *tree-shaped* if $G_{\mathcal{A}}$ is a directed tree and $\mathcal{A}$ has no multi-edges. A unary data example $(\mathcal{A}, a)$ is tree-shaped, if $\mathcal{A}$ is tree-shaped and a is the root of the tree.

Conjunctive Queries

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

$$q(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{y})$$

where $\overline{x}$ is a tuple $x_1 \cdots x_k$ of variables, called *answer variables*, $\overline{y}$ is a sequence $y_1, \ldots, y_m$ of variables, called *existential variables*, and φ is a conjunction of *concept atoms* $A(x)$ and *role atoms* $r(x, x')$ with $x, x' \in \overline{x} \cup \overline{y}$, $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. Note that the same variable can occur multiple times in $\overline{x}$. A CQ is *Boolean*, if its tuple of answer variables is empty. We use $\mathsf{var}(q)$ to refer to the set of all variables that occur in a CQ q. In a slight abuse of notation, we consider a CQ q to be a set of its atoms when convenient and write $r(x, x') \in q$, to mean that the role atom $r(x, x')$ occurs in the conjunction φ of q. Additionally, we use $R(x, x')$ for a role r to refer to the atom $r(x, x')$ if R is the role name r, and to the atom $r(x', x)$ if R is the inverse role r^-. The name *existential variables* becomes more meaningful, when we view CQs as first order logic formula. A CQ $q(\overline{x}) \leftarrow \varphi(\overline{x}, \overline{y})$ can be viewed as the first order logic formula $\psi(\overline{x}) = \exists \overline{y}\, \varphi(\overline{x}, \overline{y})$.

Motivated by the classic Chandra-Merlin theorem [CM77], we define the semantics of CQs in terms of homomorphisms. A *homomorphism* from an interpretation $\mathcal{I}$ to an interpretation $\mathcal{J}$ is a function h from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{J}}$ such that for all $A \in \mathsf{N_C}$ and $d \in A^{\mathcal{I}}$, $h(d) \in A^{\mathcal{J}}$, and for all $r \in \mathsf{N_R}$ and $(d, e) \in r^{\mathcal{I}}$, $(h(d), h(e)) \in r^{\mathcal{J}}$. For a function h from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{J}}$ we denote with $\mathsf{img}(h)$ the *image* $\{h(d) \mid d \in \Delta^{\mathcal{I}}\} \subseteq \Delta^{\mathcal{J}}$ of h. For tuples $\overline{d} = d_1 \cdots d_n$ and $\overline{e} = e_1 \cdots e_n$ we write $h(\overline{d}) = \overline{e}$ if $h(d_i) = e_i$ for all i with $1 \leq i \leq n$.

An essential fact about homomorphisms that we will use freely is that if h is a homomorphism from an interpretation $\mathcal{I}_1$ to an interpretation $\mathcal{I}_2$, and g is a homomorphism from $\mathcal{I}_2$ to an interpretation $\mathcal{I}_3$, then the composition of h and g, that is, the function h' with $h'(d) = g(h(d))$ is a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_3$. For interpretations $\mathcal{I}, \mathcal{J}$, and all tuples $\overline{d}$ over $\Delta^{\mathcal{I}}$ and all tuples $\overline{e}$ over $\Delta^{\mathcal{J}}$ we write $\mathcal{I}, \overline{d} \to \mathcal{J}, \overline{e}$ if there is a homomorphism h from $\mathcal{I}$ to $\mathcal{J}$ with $h(\overline{d}) = \overline{e}$. Since we can view ABoxes as finite interpretations, we extend this definition to ABoxes, and, for example, write $\mathcal{A}, a \to \mathcal{B}, b$ to mean $\mathcal{I}_{\mathcal{A}}, a \to \mathcal{I}_{\mathcal{B}}, b$ where $\mathcal{A}$ and $\mathcal{B}$ are ABoxes and $a \in \mathsf{ind}(\mathcal{A})$ and $b \in \mathsf{ind}(\mathcal{B})$.

With each CQ $q(\overline{x})$, we associate its *canonical data example* $(\mathcal{A}_q, \overline{x})$. The ABox $\mathcal{A}_q$ uses as individual names the variables of q, contains the concept assertion $A(x)$ for each concept atom $A(x) \in q$ and contains the role assertion $r(x, y)$ for each role atom $r(x, y) \in q$. The answer variables $\overline{x}$ of q are then used as the tuple of individual names in the data example $(\mathcal{A}_q, \overline{x})$. Similarly, we associate with each pointed ABox $(\mathcal{A}, \overline{a})$ a its canonical CQ $q_{\mathcal{A}}(\overline{a})$, that uses as variables the individual names in $\mathcal{A}$ and contains a corresponding atom for every assertion in $\mathcal{A}$. Since these two concepts are so similar, we will often view ABoxes as CQs and vice versa, via their canonical counterparts.

With a homomorphism from a CQ q to an interpretation $\mathcal{I}$, we mean a homomorphism from $\mathcal{A}_q$ to $\mathcal{I}$. Let $q(\overline{x})$ be a CQ of arity k, $\mathcal{I}$ an interpretation, and let $\overline{e}$ be an k-tuple over $\Delta^{\mathcal{I}}$. If there is a homomorphism h from $\mathcal{A}_q$ to $\mathcal{I}$ with $h(\overline{x}) = \overline{e}$, we write $q(\overline{x}) \to \mathcal{I}, \overline{e}$. If such a homomorphism exists, we say that $\overline{e}$ is an *answer of q in $\mathcal{I}$*, written as $\mathcal{I} \models q(\overline{e})$.

Let $\mathcal{O}$ be an ontology, $\mathcal{A}$ an ABox and q a CQ of arity k. We say that $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$ is an answer of q in $\mathcal{A}$ under $\mathcal{O}$ and write $\mathcal{A}, \mathcal{O} \models q(\overline{a})$ if $\mathcal{I} \models q(\overline{a})$ for every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$. This is known as the *certain answers* of q. From these definitions, it is clear that $\mathcal{A}_q, \mathcal{O} \models q(\overline{x})$ for every CQ $q(\overline{x})$ and ontology $\mathcal{O}$. Similarly to ABoxes, we say that a CQ q is satisfiable under $\mathcal{O}$ if there is a common model of $\mathcal{A}_q$ and $\mathcal{O}$. Note that, using these definitions, a CQ being unsatisfiable does not mean that it does not have any answers.

Let q and p be CQs of the same arity k. We write $q \subseteq_{\mathcal{O}} p$ and say that q *implies* p under $\mathcal{O}$ if, for all ABoxes $\mathcal{A}$ and tuples $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$,

$$\mathcal{A}, \mathcal{O} \models q(\overline{a}) \text{ implies } \mathcal{A}, \mathcal{O} \models p(\overline{a}).$$

This relationship is also referred to as *query implication* or *query containment*. We write $q \equiv_{\mathcal{O}} p$ and say that q *and* p *are equivalent* under $\mathcal{O}$ if both $q \subseteq_{\mathcal{O}} p$ and $p \subseteq_{\mathcal{O}} q$. From these definitions, it is easy to see that, for every $\mathcal{O}$, $\subseteq_{\mathcal{O}}$ is a pre-order and $\equiv_{\mathcal{O}}$ is an equivalence relation. If $q \subseteq_{\mathcal{O}} q'$ and $q \not\equiv_{\mathcal{O}} q'$ we say that q' is a *generalization* of q and that q is more specific than q'.

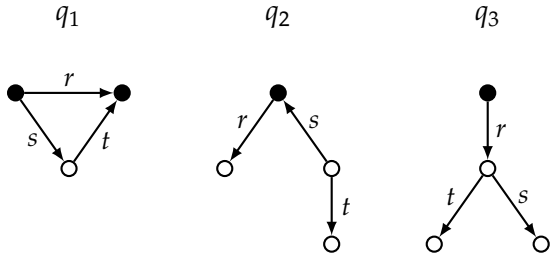

Figure 3.2: Examples of the different query classes. The queries q_1, q_2, q_3 are all CQs, only q_2, q_3 are ELIQs, and only q_3 is an ELQ. Black vertices are answer variables, white vertices are existential variables.

Example 3.1. Consider the ontology $\mathcal{O} = \{A \sqsubseteq B\}$ and the CQs

$$q_1(x_0) \leftarrow r(x_0, x_1) \wedge A(x_1) \wedge B(x_1),$$
$$q_2(x_0) \leftarrow r(x_0, x_1) \wedge A(x_1),$$
$$q_3(x_0) \leftarrow r(x_0, x_1) \wedge B(x_1).$$

Then, $q_1 \subseteq_{\mathcal{O}} q_2$ and $q_1 \subseteq_{\mathcal{O}} q_3$. Due to the concept inclusion in $\mathcal{O}$, it is also the case that $q_2 \subseteq_{\mathcal{O}} q_1$ and $q_2 \subseteq_{\mathcal{O}} q_3$. Hence, $q_1 \equiv_{\mathcal{O}} q_2$. However, $q_3 \not\subseteq_{\mathcal{O}} q_2$ and $q_3 \not\subseteq_{\mathcal{O}} q_1$. Therefore, q_3 is a generalization of q_1 and q_2.

We are further interested in two subclasses of CQs that naturally correspond to concept instance queries. A CQ $q(\overline{x})$ is acyclic, rooted, or tree-shaped if $(\mathcal{A}_q, \overline{x})$ is acyclic, rooted, or tree-shaped, respectively. We use these properties to define the following query classes:

ELIQ The $\mathcal{ELI}$ queries, or ELIQs, are the class of unary, acyclic and rooted CQs. Each $\mathcal{ELI}$ concept C naturally corresponds to an ELIQ q (and vice versa) such that $\mathcal{A}, \mathcal{O} \vDash C(a)$ if and only if $\mathcal{A}, \mathcal{O} \vDash q(a)$. For example, the $\mathcal{ELI}$ concept $C = \exists r.\top \sqcap \exists s^-.\exists t.\top$ corresponds to the ELIQ $q(x_0) \leftarrow r(x_0, x_1) \wedge s(x_2, x_0) \wedge t(x_2, x_3)$, which is displayed as q_2 in Figure 3.2.

ELQ The $\mathcal{EL}$ queries, or ELQs, are the class of unary, tree-shaped, and rooted CQs. Each $\mathcal{EL}$ concept C naturally corresponds to an ELQ q (and vice versa) such that $\mathcal{A}, \mathcal{O} \vDash C(a)$ if and only if $\mathcal{A}, \mathcal{O} \vDash q(a)$. For example, the $\mathcal{EL}$ concept $\exists r.(\exists t.\top \sqcap \exists s.\top)$ corresponds to the ELIQ $q(x_0) \leftarrow r(x_0, x_1) \wedge s(x_1, x_2) \wedge t(x_1, x_3)$, which is displayed as q_3 in Figure 3.2.

As a direct consequence of these definitions, every ELQ is an ELIQ, and every ELIQ is a CQ. Figure 3.2 shows examples of queries in these classes. The CQ q_1 is neither an ELIQ nor an ELQ, since it is not unary and contains a cycle.

For ELIQs, we adopt notation related to trees and speak about *successor* and *predecessor* variables, subtrees and leafs. In particular, we can view an ELIQ $q(x_0)$ as a tree rooted at x_0 where all edges are directed away from x_0, and each edge is labeled with a role name r or an inverse role name r^-. We then denote with q_x the ELIQ obtained from taking all atoms in the subtree below the variable x and making x the answer variable. With each variable in an ELIQ q, we associate a *codepth*, where leaf variables have codepth 0, and non-leaf variables have the codepth that is the minimum of their successors plus one.

Example 3.2. Consider the ELIQ

$$q(x_0) \leftarrow r(x_0, x_1) \wedge s(x_0, x_3) \wedge A(x_1) \wedge s(x_2, x_1).$$

The role atoms $r(x_0, x_1)$ and $s(x_0, x_3)$ are directed away from the answer variable x_0 and the role atom $s(x_2, x_1)$ is directed towards x_0. We can also view $s(x_2, x_1)$ as an s^- atom, that is directed away from x_0. Furthermore, the ELIQ q_{x_1}, that is the subtree rooted at x_1, is

$$q_{x_1}(x_1) \leftarrow A(x_1) \wedge s(x_2, x_1).$$

The codepth of x_2 in q is 0, the codepth of x_1 is 1, and the codepth of x_0 is 2.

A *signature* Σ is a set of concept and role names. For a query class $\mathcal{Q}$, we denote with $\mathcal{Q}_\Sigma$ the class of all queries from $\mathcal{Q}$ that only use symbols in Σ. Note that all query classes $\mathcal{Q}$ we have defined are *infinite*, and, assuming that Σ contains a role name, all $\mathcal{Q}_\Sigma$ are infinite as well. For any syntactic object o (like concepts, ontologies, ABoxes or queries), we use $\mathsf{sig}(o)$ to denote the set of concept and role names used in o, and $\|o\|$ to denote the *size* of o, that is, the number of symbols needed to write o as a word encoded over a finite fixed alphabet, where each occurrence of concept or role names contributes one symbol.

The Direct Product

Closely related to conjunctive queries and homomorphisms is the operation of constructing a direct product of two interpretations. The *direct product* of two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ is the interpretation $\mathcal{I}_1 \times \mathcal{I}_2$ defined by setting

$$\begin{aligned}
\Delta^{\mathcal{I}_1 \times \mathcal{I}_2} &= \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}, \\
A^{\mathcal{I}_1 \times \mathcal{I}_2} &= A^{\mathcal{I}_1} \times A^{\mathcal{I}_2}, \text{ for all } A \in \mathsf{N_C}, \\
r^{\mathcal{I}_1 \times \mathcal{I}_2} &= \{((d_1, d_2), (e_1, e_2)) \mid (d_i, e_i) \in r^{\mathcal{I}_i} \text{ for } i \in \{1, 2\}\}, \text{ for all } r \in \mathsf{N_R}.
\end{aligned}$$

Let $\overline{a} = a_1 \dots a_n$ and $\overline{b} = b_1 \dots b_n$ be tuples of the same arity. With $\overline{a} \otimes \overline{b}$ we denote the tuple $(a_1, b_1) \dots (a_n, b_n)$. From the definition of direct products, it is clear that

$|\Delta^{\mathcal{I}_1 \times \mathcal{I}_2}| = |\Delta^{\mathcal{I}_1}| \cdot |\Delta^{\mathcal{I}_2}|$. Hence, the product of finite interpretations can be computed in polynomial time. The important standard properties of direct products are summarized in the following lemma.

Lemma 3.3. *Let $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{J}$ be interpretations and $\bar{e}_1$, $\bar{e}_2$ and $\bar{e}$ tuples. Then, the following are equivalent.*

1. $\mathcal{J}, \bar{e} \to \mathcal{I}_1, \bar{e}_1$ *and* $\mathcal{J}, \bar{e} \to \mathcal{I}_2, \bar{e}_2$;
2. $\mathcal{J}, \bar{e} \to \mathcal{I}_1 \times \mathcal{I}_2, \bar{e}_1 \otimes \bar{e}_2$.

The same applies to ABoxes and CQs viewed as finite interpretations. A direct consequence of Lemma 3.3 is that $\mathcal{I}_1 \models q(\bar{e}_1)$ and $\mathcal{I}_2 \models q(\bar{e}_2)$ if and only if $\mathcal{I}_1 \times \mathcal{I}_2 \models q(\bar{e}_1 \otimes \bar{e}_2)$.

Reasoning

Associated with ontologies and queries are certain decision problems, referred to as *reasoning tasks*, that involve reasoning with the logical statements in the ontology. Their computational complexity varies, depending on the query class and the ontology language. Here, we briefly introduce the main reasoning tasks that are relevant for learning queries under ontologies, and comment on their *combined complexity*.

The first basic reasoning task is *satisfiability*: Given an ABox $\mathcal{A}$ and an ontology $\mathcal{O}$, decide whether $\mathcal{A}$ is satisfiable under $\mathcal{O}$. Conjunctive, $\mathcal{EL}^r$ and $\mathcal{ELI}$ ontologies do not contain disjointness or functionality constraints and every ABox is therefore satisfiable under those ontologies. Ontologies written in $\mathcal{ELIHF}_\bot$ may constraints, and deciding satisfiability of ABoxes is ExpTime-complete [KRH13]. For *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$, *DL-Lite*$_{\text{horn}}$, *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$, and *DL-Lite*$_{\text{core}}$ ontologies, satisfiability can be decided in P [Art+09].

Another basic reasoning task is *concept subsumption*: Given as input an ontology $\mathcal{O}$, and concepts C, D, decide whether $\mathcal{O} \models C \sqsubseteq D$. In ontology languages with concept disjointness constraints, this can be decided by answering whether C viewed as an ABox is not satisfiable under $\mathcal{O} \cup \{C \sqcap D \sqsubseteq \bot\}$. Therefore, deciding concept subsumption is in P for *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$, *DL-Lite*$_{\text{horn}}$, *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ and *DL-Lite*$_{\text{core}}$, and in ExpTime for $\mathcal{ELIHF}_\bot$. For ontology languages without concept disjointness constraints, separate reasoning algorithms are necessary to decide concept subsumption. For conjunctive or $\mathcal{EL}$ ontologies, concept subsumption can be decided in P [Bra04]. For $\mathcal{ELI}$ ontologies, deciding concept subsumption is ExpTime-complete [BBL08], and therefore the same is true for $\mathcal{ELIHF}_\bot$ ontologies.

Since we are interested in querying under ontologies, the computational complexity of *query answering under ontologies* are also relevant: Given an ontology $\mathcal{O}$,

Table 3.2: Summary of the (combined) complexity of various reasoning tasks

$\mathcal{L}$	Satisfiability	$\mathcal{O} \models C \sqsubseteq D$	$\mathcal{A}, \mathcal{O} \models q_{ELIQ}(\overline{a})$	$\mathcal{A}, \mathcal{O} \models q_{CQ}(\overline{a})$
$\mathcal{ELIHF}_\bot$	ExpTime-c	ExpTime-c	ExpTime-c	ExpTime-c
$\mathcal{ELI}$	trivial	ExpTime-c	ExpTime-c	ExpTime-c
$\mathcal{EL}$	trivial	in P	in P	NP-c
DL-Lite$^{\mathcal{F}}_{\text{horn}}$	in P	in P	in NP	NP-c
DL-Lite$_{\text{horn}}$	in P	in P	in NP	NP-c
DL-Lite$^{\mathcal{HF}}_{\text{core}}$	in P	in P	NP-c	NP-c
DL-Lite$_{\text{core}}$	in P	in P	in P	NP-c
$\varnothing$	trivial		in P	NP-c

a CQ q, and a data example $(\mathcal{A}, \overline{a})$, decide whether $\mathcal{A}, \mathcal{O} \models q(\overline{a})$. Already in the case without an ontology, answering CQs is NP-complete [CM77]. For ontologies written in more expressive DLs such as $\mathcal{ELI}$ and $\mathcal{ELIHF}_\bot$, answering CQs is ExpTime-complete [Eit+08]. However, for ontologies written in $\mathcal{EL}$ or DLs of the *DL-Lite* family, where concept subsumption is in P, answering CQs remains NP-complete [Cal+05; Ros07].

The complexity of query answering improves when we restrict q to be an ELIQ. In fact, acyclicity of q suffices. Then, answering an ELIQ is in P in the case without ontologies [Yan81]. Moreover, it remains in P for $\mathcal{EL}$ and *DL-Lite*$_{\text{core}}$ ontologies [Bie+13]. The role inclusions in *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ ontologies unfortunately make answering ELIQs NP-complete [KKZ11]. For ontologies written $\mathcal{ELI}$ and $\mathcal{ELIHF}_\bot$ the ExpTime-complete concept subsumption dominates the complexity of query answering, and ELIQ answering is not easier than CQ answering.

All mentioned complexity results are summarized in Table 3.2. When investigating query answering under ontologies, one is commonly also interested in *data complexity*, that is, the computational complexity of query answering when the ontology and the query are assumed to be fixed. Low data complexity for query answering is one of the main motivations of the *DL-Lite* family of DLs. However, data complexity is not so relevant for us, as we will not assume that ontologies or specific queries are fixed when learning queries.

Universal Models

An important property of $\mathcal{ELIHF}_\bot$ and all of its sublanguages is that they are contained in the Horn fragment of first order logic. This gives $\mathcal{ELIHF}_\bot$ ontologies the *universal model property*, meaning that they possess a single model that is homomorphically contained in all of their models. Next, we review the construction of

such models and their properties.

For this, it is useful to assume that ontologies are in *normal form*. An $\mathcal{ELIHF}_\bot$ ontology is in *normal form* if all concept inclusions in it are of one of the following forms

$$A_1 \sqsubseteq \exists R.A_2 \qquad \exists R.A_1 \sqsubseteq A_2 \qquad A_1 \sqcap A_2 \sqsubseteq A_3$$

where A_1, A_2, A_3 are concept names or $\top$ and R is a role. It is well known that every $\mathcal{ELIHF}_\bot$ ontology can be converted into normal form in polynomial time, by introducing additional concept names [Baa+17].

For a set M of concept names, we write $\bigsqcap M$ as a shorthand for $\bigsqcap_{A\in M} A$. Let $\mathcal{A}$ be an ABox and $\mathcal{O}$ an $\mathcal{ELIHF}_\bot$ ontology. We define the interpretation $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ as follows. For $a \in \mathsf{ind}(\mathcal{A})$, sets of concept names M, M', and a role R, we write $a \rightsquigarrow^R_{\mathcal{A},\mathcal{O}} M$ if

1. $\mathcal{A},\mathcal{O} \vDash \exists R.\bigsqcap M(a)$ and M is maximal with this condition, and
2. there is no $b \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{A},\mathcal{O} \vDash R(a,b)$ and $\mathcal{A},\mathcal{O} \vDash \bigsqcap M(b)$.

We write $M \rightsquigarrow^R_{\mathcal{O}} M'$ if $\mathcal{O} \vDash \bigsqcap M \sqsubseteq \exists R.\bigsqcap M'$ and M' is maximal with this condition.

A *trace* for $\mathcal{A}$ and $\mathcal{O}$ is a sequence $aR_1M_1 \cdots R_nM_n$ for $n \geq 0$, where $a \in \mathsf{ind}(\mathcal{A})$, $R_1, \dots, R_n$ are roles that occur in $\mathcal{O}$, and $M_1, \dots, M_n$ are sets of concept names that occur in $\mathcal{O}$ such that

1. $a \rightsquigarrow^{R_1}_{\mathcal{A},\mathcal{O}} M_1$, and
2. $M_i \rightsquigarrow^{R_{i+1}}_{\mathcal{O}} M_{i+1}$ and $\mathsf{func}(R_i^-) \in \mathcal{O}$ implies $R_{i+1} \neq R_i^-$, for $1 \leq i < n$.

We say that a trace $aR_1M_1 \cdots R_nM_n$ starts with $a \in \mathsf{ind}(\mathcal{A})$ and has length n. We call a trace *proper* if it has length at least one, that is, it is not an element of $\mathsf{ind}(\mathcal{A})$.

The set $\mathbf{T}$ of all traces for $\mathcal{A}$ and $\mathcal{O}$ forms $\Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. We define $\cdot^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ for all $A \in \mathsf{N_C}$ and all $r \in \mathsf{N_R}$ as follows.

$$\begin{aligned}
A^{\mathcal{U}_{\mathcal{A},\mathcal{O}}} =& \{a \in \mathsf{ind}(\mathcal{A}) \mid \mathcal{A},\mathcal{O} \vDash A(a)\} \cup \\
& \{tRM \in \mathbf{T} \mid A \in M\}, \\
r^{\mathcal{U}_{\mathcal{A},\mathcal{O}}} =& \{(a,b) \in \mathsf{ind}(\mathcal{A})^2 \mid \mathcal{A},\ \mathcal{O} \vDash r(a,b)\} \cup \\
& \{(t, tsM) \mid tsM \in \mathbf{T},\ \mathcal{O} \vDash s \sqsubseteq r\} \cup \\
& \{(ts^-M, t) \mid ts^-M \in \mathbf{T},\ \mathcal{O} \vDash s \sqsubseteq r\}.
\end{aligned}$$

Example 3.4. Consider the ABox $\mathcal{A} = \{r(a,b)\}$ and the ontology

$$\mathcal{O} = \{A \sqsubseteq \exists r.A, B \sqsubseteq s.B, r \sqsubseteq s, \mathsf{func}(r)\}.$$

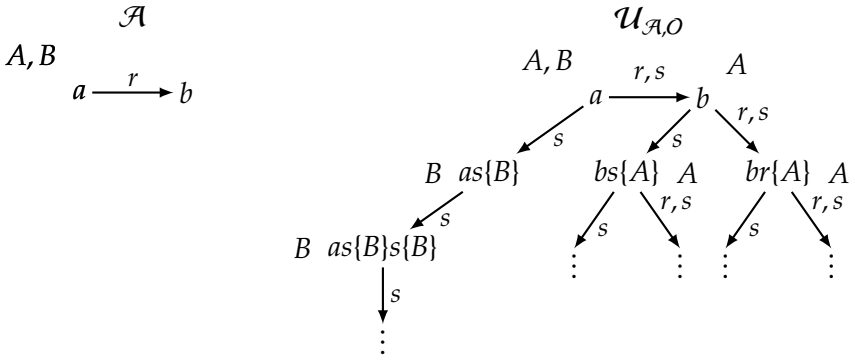

Figure 3.3: The ABox $\mathcal{A}$ and the infinite interpretation $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ for $\mathcal{A} = \{r(a,b)\}$ and $\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists s.B,\ r \sqsubseteq s,\ \mathsf{func}(r)\}$.

An initial segment of the interpretation $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is displayed in Figure 3.3. In $\mathcal{U}_{\mathcal{A},\mathcal{O}}$, the trace $as\{B\}$ is an s-successor of a, as $a \rightsquigarrow^{s}_{\mathcal{A},\mathcal{O}} \{B\}$. Note that $a \not\rightsquigarrow^{s}_{\mathcal{A},\mathcal{O}} \{A\}$ and $a \not\rightsquigarrow^{r}_{\mathcal{A},\mathcal{O}} \{A\}$ due to the $r(a,b)$ assertion in $\mathcal{A}$. Since $\mathcal{A},\mathcal{O} \models A(b)$, $b \rightsquigarrow^{s}_{\mathcal{A},\mathcal{O}} \{A\}$ and $b \rightsquigarrow^{r}_{\mathcal{A},\mathcal{O}} \{A\}$. Therefore, the traces $bs\{A\}$ and $br\{A\}$ are attached to b.

Note that $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is usually infinite. To each element $a \in \mathsf{ind}(\mathcal{A})$, the traces starting with a are attached in a tree-like structure. The traces do not necessarily form an acyclic interpretation, as role inclusions can introduce multi-edges, but their underlying graph is acyclic. Therefore, we also use tree terminology when talking about traces. Commonly, we use *the subtree below a* or *the subtree attached to a* to mean the restriction of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to the traces starting with a.

The interpretation $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is then a *universal model* of $\mathcal{A}$ and $\mathcal{O}$, the properties of which are made precise by the following lemma. Its proof is standard, see, for example, [Bot+16].

Lemma 3.5. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form and $\mathcal{A}$ an ABox that is satisfiable under $\mathcal{O}$. Then*

1. *$\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is a model of $\mathcal{A}$ and $\mathcal{O}$;*
2. *for every model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}$, there is a homomorphism h from $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to $\mathcal{I}$ with $h(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{ind}(\mathcal{A})$.*
3. *for all k-ary CQs $q(\overline{x})$ and all $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$, $\mathcal{A},\mathcal{O} \models q(\overline{a})$ if and only if $\mathcal{U}_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$.*

We call a model that fulfills Point 3 of Lemma 3.5 *CQ-universal*. Occasionally, we will also use models that are $\mathcal{Q}$-universal for some query class $\mathcal{Q}$ that are not CQs and mean that they fulfill Point 3 not for all CQs, but for all queries in $\mathcal{Q}$.

The definition of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is very close to standard definitions of universal models for $\mathcal{ELIH}_\bot$ [BO15], that is, $\mathcal{ELIHF}_\bot$ without functionality constraints, with additional considerations to accommodate these functionality constraints. Compare it also to definitions of universal models for $\mathcal{ELIHF}_\bot$ without the restriction on the interaction of role inclusions and functionality constraints, such as in [LP22]. Due to the restriction, we can use single roles instead of sets of roles in our definition of traces. Note that this means that $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ is not a model of $\mathcal{ELIHF}_\bot$ ontologies in which the interaction of functionality constraints and role inclusions is not restricted.

Example 3.6. Consider the ABox $\mathcal{A} = \{A(a)\}$ and the ontology

$$\mathcal{O} = \{A \sqsubseteq \exists r_1.\top,\ A \sqsubseteq \exists r_2.\top,\ r_1 \sqsubseteq s,\ r_2 \sqsubseteq s,\ \mathsf{func}(s)\},$$

which is not an $\mathcal{ELIHF}_\bot$ ontology according to the definition in Section 3.1, as $\mathsf{func}(s) \in \mathcal{O}$, but there is a role r_1 with $r_1 \sqsubseteq s \in \mathcal{O}$. Then, $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ contains the traces $ar_1\varnothing$ and $ar_2\varnothing$ with $(a, ar_1\varnothing), (a, ar_2\varnothing) \in s^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$, which violates the functionality constraint $\mathsf{func}(s) \in \mathcal{O}$.

We will often consider models $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ where $\mathcal{A}$ is a CQ q viewed as an ABox $\mathcal{A}_q$. In these cases, we will write $\mathcal{U}_{q,\mathcal{O}}$ instead of $\mathcal{U}_{\mathcal{A}_q,\mathcal{O}}$. The following lemma is a direct consequence of Point 3 of Lemma 3.5 and the homomorphism-based semantics of CQs.

Lemma 3.7. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form and $p(\overline{y})$, $q(\overline{x})$ CQs such that p is satisfiable under $\mathcal{O}$. Then, $p \subseteq_{\mathcal{O}} q$ if and only if $\mathcal{A}_p, \mathcal{O} \models q(\overline{y})$ if and only if $\mathcal{U}_{p,\mathcal{O}} \models q(\overline{y})$.*

Occasionally, when composing homomorphisms, we will need to extend the domain of homomorphisms from queries to their universal model.

Lemma 3.8. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form, $\mathcal{A}$ an ABox, $q(\overline{x})$ a CQ, and $\overline{a}$ a tuple over $\mathsf{ind}(\mathcal{A})$. Every homomorphism h from q to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ with $h(\overline{x}) = \overline{a}$ can be extended to a homomorphism h' from $\mathcal{U}_{q,\mathcal{O}}$ to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ with $h'(\overline{x}) = \overline{a}$.*

The proofs of these lemmas are standard and omitted.

3.2 The Fitting Problem

In the fitting model of learning, the learning algorithm receives labeled data examples as input and must compute a query that fits the labeled examples. Formally, a *labeled data example* is an example $(\mathcal{A}, \overline{a})$ together with the label + (*positive example*) or − (*negative example*). A CQ q *fits a collection of labeled examples* E under an ontology $\mathcal{O}$ if

- $\mathcal{A}, \mathcal{O} \models q(\bar{a})$ for all $(\mathcal{A}, \bar{a}, +) \in E$, and
- $\mathcal{A}, \mathcal{O} \not\models q(\bar{a})$ for all $(\mathcal{A}, \bar{a}, -) \in E$.

Observe that E is not a set, that is, E is allowed to contain duplicate data examples. This is not essential here, but makes the definition of fitting closer to PAC learning. The decision problem at the core of computing a fitting query given examples is as follows.

Definition 3.9 (Fitting problem)**.** Let $\mathcal{Q}$ be a query class and $\mathcal{L}$ an ontology language. The *fitting problem* for $\mathcal{Q}$ and $\mathcal{L}$ is the problem to decide, given an $\mathcal{L}$ ontology $\mathcal{O}$, a signature Σ, and a set of labeled examples E, whether there is a query $q \in \mathcal{Q}_\Sigma$ that fits E under $\mathcal{O}$.

The fitting problem is also referred to as the *separability problem*, the *weak separability problem*, or *query-by-example* [Fun+19; GJS18; Jun+20]. The name *weak separability* refers to the contrast to *strong separability*. In the strong separability problem, it is required that $\mathcal{B}, \mathcal{O} \models \neg q(\bar{b})$ for all negative examples $(\mathcal{A}, \bar{a}, -) \in E$. In our setting, this does not make sense since $\mathcal{B}, \mathcal{O} \models \neg q(\bar{b})$ is never true due to the open world assumption and choice of ontology language. However, for stronger ontology languages that can express negation, like $\mathcal{ALC}$, strong separability may be desired. See for example [BN00; Fan+18; Leh+14] for works on strong separability. Since we only consider lightweight description logics where reasoning is possible in polynomial time, only fitting in the above sense of weak separability is useful to us.

As it lies at the core of many query engineering problems, the complexity of the fitting problem has been investigated both in the setting of databases without ontologies and in the setting of knowledge bases.

Willard showed that computing a fitting CQ, even under the empty ontology, is coNExpTime-complete [tCD15; Wil10]. Part of this high complexity lies in the fact that the size of a fitting CQ may be exponential in the size of the examples. In the case of ELQs, the size of a fitting query may even be double-exponential. Fortunately, the existence of a fitting ELQ (or ELIQ) under the empty ontology can be decided in ExpTime based on a simulation-based characterization, and is indeed only ExpTime-complete [BR17]. The same is true for deciding the existence of fitting ELQs under $\mathcal{ELI}$ ontologies [Fun+19], as finite models, although of exponential size, of $\mathcal{ELI}$ ontologies that are ELQ-universal exist. Since all relevant features of *DL-Lite*$_{\text{horn}}$ are available in $\mathcal{ELI}$, this also applies to ELQs under *DL-Lite*$_{\text{horn}}$ ontologies.

Finite ELIQ- or CQ-universal models of $\mathcal{EL}$ or $\mathcal{ELI}$ ontologies do not exist. Nevertheless, under Horn-$\mathcal{ALC}$ ontologies (which includes all $\mathcal{EL}$ ontologies), the fitting

Table 3.3: Known complexity results of the fitting problem for $\mathcal{Q}$ and $\mathcal{L}$

$\mathcal{L}$	ELQ	ELIQ	CQ
$\mathcal{ELI}$	ExpTime-c	undecidable	undecidable
$\mathcal{EL}$	ExpTime-c	ExpTime-h	coNExpTime-c
DL-Lite$_{\text{core}}$ / *DL-Lite*$_{\text{horn}}$	ExpTime-c	ExpTime-h	coNExpTime-h
$\varnothing$	ExpTime-c	ExpTime-c	coNExpTime-c

problem for CQs is not harder than in the case without ontologies and can be decided in coNExpTime [GJS18]. For ELIQs under $\mathcal{EL}$ ontologies, the exact complexity is not known, but the ExpTime-hardness from the case under the empty ontology transfers. Unfortunately, under $\mathcal{ELI}$ ontologies, the fitting problem for ELIQs and CQs even becomes undecidable [Fun+19]. This is slightly worrying, as therefore no algorithm that searches for fitting ELIQs under $\mathcal{ELI}$ ontologies can be complete and terminating. However, exact learning and PAC learning algorithms work under the assumption that a fitting query exists and therefore do not necessarily need to solve an undecidable problem.

For DLs of the *DL-Lite* family, only few bounds on the complexity of the fitting problem for CQs or ELIQs are known, as existing results mostly focus on UCQs [CCL21; Ort19]. In general, deciding the existence of a fitting UCQ is computationally simpler than the existence of a fitting CQ, as a fitting query for each positive example can be determined separately, and then joined in a disjunction. This makes the fitting problem for UCQs under the empty ontology only coNP-complete.

Table 3.3 summarizes the mentioned results. It is interesting to note that the size of a fitting query is not the only source of the high complexities of the fitting problems. These problems remain at least NP-hard, even if algorithms take a bound on the size of the fitting query as input [tCat+24]. We revisit this in Chapter 6.

For learning queries, we are not only interested in deciding if a fitting query exist, but also wish to obtain a fitting query. For this, we use the notion of *fitting algorithms.*

Definition 3.10 (Fitting algorithm)**.** Let $\mathcal{Q}$ be a query class and $\mathcal{L}$ an ontology language. A *fitting algorithm* for $\mathcal{Q}$ and $\mathcal{L}$ is an algorithm that takes as input a signature Σ, an $\mathcal{L}$ ontology $\mathcal{O}$ with $\mathsf{sig}(\mathcal{O}) \subseteq \Sigma$, and a set of labeled examples E, and returns a query $q \in \mathcal{Q}_\Sigma$ that fits E under $\mathcal{O}$ if such a query exists.

Of course, fitting algorithms according to Definition 3.10 are not algorithms in the usual sense, as it is not required to terminate if no fitting query exists in $\mathcal{Q}_\Sigma$. Additionally, fitting algorithms may return any query if no fitting query exists. We

chose this definition for uniformity with the other learning models, where a fitting query is guaranteed to exist. These issues do not occur in practice, as for many combinations of $\mathcal{Q}$ and $\mathcal{L}$ there are upper bounds on the size of a fitting query, like the aforementioned bounds on ELQs and CQs, which can be used to ensure that a fitting algorithm in the sense of Definition 3.10 always terminates. Furthermore, as query answering is usually computationally easier than the fitting problem, it can in a second step be verified that the result of a fitting algorithm in the sense of Definition 3.10 really fits the input examples.

3.3 Exact Learning

Angluin's *exact learning* framework [Ang88b] models learning as an interactive process involving two parties, a *learner* and a *teacher*. It was conceived in the context of learning finite automata from word examples in polynomial time [Ang87]. We informally described this framework in Chapter 1. In this section, we define it formally, specialized to our setting of learning queries under ontologies.

Both the learner and the teacher agree on a query class $\mathcal{Q}$ and know an ontology $\mathcal{O}$, as well as a signature Σ, but only the teacher knows a target query $q_T \in \mathcal{Q}_\Sigma$. The learner then attempts to identify q_T by posing two different kinds of questions to the teacher that we already saw in Chapter 1:

Membership query Given an example $(\mathcal{A}, \overline{a})$, the teacher responds with *yes* if $\mathcal{A}, \mathcal{O} \models q_T(\overline{a})$ and *no* otherwise.

Equivalence query Given a hypothesis query $q_H \in \mathcal{Q}_\Sigma$, the teacher responds *yes* if $q_H \equiv_\mathcal{O} q_T$. If this is not the case, the teacher produces a counterexample, that is, an example $(\mathcal{A}, \overline{a})$ such that $\mathcal{A}, \mathcal{O} \models q_T(\overline{a})$ and $\mathcal{A}, \mathcal{O} \not\models q_H(\overline{a})$ or vice versa.

Note that in membership queries, $\mathsf{sig}(\mathcal{A})$ is unrestricted.

It may be confusing that membership queries and equivalence queries are called queries when we already aim to learn conjunctive queries. These two sorts of queries are not related. Sometimes membership queries are renamed to *membership oracle calls* to avoid this confusion [tCD22], but we choose to keep the name membership query, since it is established in the exact learning literature. Similarly, exact learning is often concerned with learning *concepts*, which are defined to be sets of examples, and not syntactic objects, like the description logic concepts defined in Section 3.1. We mostly avoid this clash of terms by defining our learning models for query classes and not for abstract classes of concept.

In both membership queries and equivalence queries, the teacher must respond correctly and truthfully, but the counterexamples provided by the teacher do not

have to be helpful. In fact, the learner must be able to identify the target query even if the teacher acts as an adversary.

In the exact learning framework, we desire a *learning algorithm* for the learner to execute that identifies q_T in all cases.

Definition 3.11 (Exact learning algorithm)**.** Let $\mathcal{Q}$ be a query class and $\mathcal{L}$ an ontology language. An *exact learning algorithm* for $\mathcal{Q}$ under $\mathcal{L}$ ontologies is an algorithm $\mathbf{A}$ that, for all signatures Σ, all $\mathcal{L}$ ontologies $\mathcal{O}$ with $\mathrm{sig}(\mathcal{O}) \subseteq \Sigma$, and all $q_T \in \mathcal{Q}_\Sigma$, when started with input $\mathcal{O}$ and Σ, and allowed to ask membership and equivalence queries that are answered with regard to q_T, $\mathbf{A}$ returns a hypothesis $q_H \in \mathcal{Q}_\Sigma$ with $q_H \equiv_\mathcal{O} q_T$.

If there exists an exact learning algorithm for $\mathcal{Q}$ under $\mathcal{L}$ ontologies, then $\mathcal{Q}$ is *exactly learnable* under $\mathcal{L}$ ontologies.

Using this definition, every enumerable query class is exactly learnable under every ontology language. This is because there always is a learning algorithm that enumerates all queries in $\mathcal{Q}_\Sigma$ ordered by size and returns the first query for which the teacher responds positively to an equivalence query. For the query classes we are interested in, this means that the algorithm makes a number of equivalence queries that is exponential in $\|q_T\|$, as there is an exponential number of queries that are smaller than q_T. This is often not practical, especially when equivalence queries need to be answered by a human teacher. Hence, we are interested in notions of *efficient exact learnability*, where the learner is not permitted to make this exponential number of equivalence queries.

Polynomial time learnability The first notion of efficiency restricts the algorithm to run in polynomial time.

The query class $\mathcal{Q}$ is *polynomial time learnable* under $\mathcal{L}$ ontologies, if there is an exact learning algorithm $\mathbf{A}$ for $\mathcal{Q}$ under $\mathcal{L}$ ontologies, and there is a polynomial $p(n_\Sigma, n_\mathcal{O}, n_{q_T}, n_\mathcal{A})$ such that at *each point* during the run of $\mathbf{A}$, the time used by $\mathbf{A}$ is bounded by $p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_\mathcal{A})$, where $n_\mathcal{A}$ is the size of the largest counterexample returned by an equivalence query so far.

Note that this notion is slightly more restrictive than demanding the total running time of $\mathbf{A}$ to be bounded by $p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_\mathcal{A})$. This is because we want to allow p to depend on $n_\mathcal{A}$, in order to give $\mathbf{A}$ enough time to read large counterexamples whose size is not bounded by the other parameters. But in certain scenarios bounding the total running time this way could be abused by learning algorithms: $\mathbf{A}$ could first perform an exponential search for q_T as described above, and then, with knowledge of q_T, ask an equivalence query that forces the teacher to produce an exponentially sized counterexample, thereby sufficiently raising the running time bound [Ang90].

Polynomial query learnability For some choices of $\mathcal{Q}$ and $\mathcal{L}$, polynomial time learnability seems implausible due to reasons not directly linked to the learning itself. For example, when learning under $\mathcal{ELI}$ ontologies, ExpTime-complete reasoning problems may need to be solved by the learner. In these cases, we might relax the bound on the running time of the learning algorithm, but still demand that the number of membership queries and equivalence queries is bounded to forbid exhaustive search strategies.

A query class $\mathcal{Q}$ is *polynomial query learnable* under $\mathcal{L}$ ontologies, if there is an exact learning algorithm $\mathbf{A}$ for $\mathcal{Q}$ under $\mathcal{L}$ ontologies, and there is a polynomial $p(n_\Sigma, n_\mathcal{O}, n_{q_T}, n_\mathcal{A})$ such that at each point during the run of $\mathbf{A}$, the sum of the sizes of the inputs to membership and equivalence queries up to that point is bounded by $p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_\mathcal{A})$ and every query returned by $\mathbf{A}$ has size at most $p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_\mathcal{A})$.

The bound on the size of the output of $\mathbf{A}$ is non-standard in this definition. Without it, a learning algorithm could identify q_T using membership queries and equivalence queries of size bounded by $p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_\mathcal{A})$, and then return a query equivalent to q_T, but exponentially larger. This is not desirable, since it breaks a connection to PAC learning, which we describe in Section 3.4.

It follows directly from these definitions that every query class $\mathcal{Q}$ that is polynomial time learnable under $\mathcal{L}$ ontologies is also polynomial query learnable. Additionally, polynomial time learnability and polynomial query learnability are anti-monotone in $\mathcal{L}$: If every $\mathcal{L}'$ ontology is also a $\mathcal{L}$ ontology and $\mathcal{Q}$ is polynomial time learnable under $\mathcal{L}$ ontologies, then $\mathcal{Q}$ is also polynomial time learnable under $\mathcal{L}'$ ontologies using the same learning algorithm. This makes the case of learning without ontologies a special case of learning under ontologies. The same is not true for the query class $\mathcal{Q}$: If $\mathcal{Q}' \subseteq \mathcal{Q}$ and $\mathcal{Q}$ is polynomial time learnable under $\mathcal{L}$ ontologies, then $\mathcal{Q}'$ is not necessarily polynomial time learnable under $\mathcal{L}$ ontologies. This is because the hypotheses used in equivalence queries must then be from $\mathcal{Q}'$, but the original learning algorithm produces hypotheses from $\mathcal{Q}$.

In Definition 3.11 we assume that $\mathsf{sig}(\mathcal{O}) \subseteq \Sigma$. This assumption gives more freedom to learning algorithms, as they can freely use all symbols in the ontology as part of equivalence queries.

In some formulations of the exact learning model, the learner has access to more kinds of questions than membership queries and equivalence queries [Ang88b]. One kind that is interesting in our setting are *subset queries*.

Subset query Given query $q_H \in \mathcal{Q}_\Sigma$, the teacher answers with *yes* if $q_H \subseteq_\mathcal{O} q_T$ and with *no* otherwise.

For all query classes that are subclasses of CQs, subset queries do not make learning algorithms more powerful. The reason for this is Lemma 3.7. Consider a CQ $q(\overline{x})$ and the corresponding example $(\mathcal{A}_q, \overline{x})$. Then, the response to a subset query with q is positive if and only if $q \subseteq_{\mathcal{O}} q_T$. This in turn is true if and only if $\mathcal{A}_q, \mathcal{O} \models q_T(\overline{x})$, which is precisely what is answered by a membership query. Hence, one can replace uses of subset queries with uses of membership queries.

However, the other direction, replacing membership queries with subset queries, is not always possible. One can replace a membership query with example $(\mathcal{A}, \overline{a})$ with a subset query, only if $q_{\mathcal{A}} \in \mathcal{Q}$. That is, if $\mathcal{Q}$ = ELQ, then a membership query with the example $(\{r(a,a)\}, a)$ cannot be replaced with a subset query, as $q_{\mathcal{A}}$ is not an ELQ.

It is, of course, also of interest to show that some combinations of query class and ontology language are not efficiently learnable. Angluin introduced several techniques to show lower bounds for the number of membership queries and equivalence queries that every (correct) learning algorithm has to use in the worst case. A basic technique for showing lower bounds for membership queries is the following.

Lemma 3.12 ([Ang88b])**.** *Let $\mathcal{L}$ be an ontology language and $\mathcal{Q}$ a query class. If there is a set S of $\mathcal{Q}$ queries and an $\mathcal{L}$ ontology such that for all examples $(\mathcal{A}, \overline{a})$ and for all $p, q \in S$ with $p \neq q$,*

$$\mathcal{A}, \mathcal{O} \models q(\overline{a}) \text{ and } \mathcal{A}, \mathcal{O} \models p(\overline{a}) \text{ implies } \mathcal{A}, \mathcal{O} \models q'(\overline{a}) \text{ for all } q' \in S,$$

then every learning algorithm that uses only membership queries to learn $\mathcal{Q}$ queries under $\mathcal{L}$ ontologies must make at least $|S| - 1$ membership queries in the worst case.

Proof. To show this, we view the teacher as an *adversary*, who tries to avoid identification of a target query $q_T \in S$ as long as possible. For this, we will describe a strategy for the teacher to maintain the set S as a set of possible target queries that is consistent with all answers given to membership queries so far. The properties of S make sure that the teacher needs to only remove at most one query from S for each membership query that the learner asks, meaning that for every learning algorithm that asks fewer than $|S| - 1$ membership queries, two candidates remain in S that the learner cannot distinguish.

Assume to the contrary, that there is a learning algorithm that can identify every single choice of target query from S with fewer than $|S| - 1$ membership queries. If the learner asks a membership query with example $(\mathcal{A}, \overline{a})$, the adversarial teacher responds as follows:

1. if there is no $q \in S$ such that $\mathcal{A}, \mathcal{O} \models q(\overline{a})$, answer *no*;
2. if $\mathcal{A}, \mathcal{O} \models q(\overline{a})$ for a single $q \in S$, answer *no* and remove q from S;

3. if $\mathcal{A}, \mathcal{O} \models q(\overline{a})$ for more than one $q \in S$, answer *yes*.

Note that in every case, the queries that remain in S are consistent with the answers given so far. In the third case, this is due to the property of S that if $\mathcal{A}, \mathcal{O} \models q(\overline{a})$ and $\mathcal{A}, \mathcal{O} \models p(\overline{a})$ for $p, q \in S$ with $p \neq q$, then $\mathcal{A}, \mathcal{O} \models q'(\overline{a})$ for all $q' \in S$. Thus, the learner cannot distinguish the remaining candidate queries in S based on the answers so far, and we have arrived at a contradiction. □

We will use several extensions of this technique to show that certain query classes cannot be learned using only a polynomial number of membership queries.

Beyond Lemma 3.12, many more sophisticated combinatorial approaches have been developed to show lower bounds on the required number of membership queries and equivalence queries, like *certificates* [Hel+96], the *generalized teaching dimension* [Heg95], or the *Littlestone dimension* [CF20] See [Ang04] for a summary of these techniques. Unfortunately, no lower bounds in the area of learning queries under ontologies have been established using these techniques so far.

3.4 Probably Approximately Correct Learning

The model of *probably approximately correct* (*PAC*) *learning* was introduced by Valiant in the context of learning Boolean functions [Val84], although not under this name. The name was later coined by Angluin [Ang88a]. In this section, we define the PAC model for learning queries under ontologies.

At the core of the PAC model lies the assumption that data examples are independently drawn from an unknown probability distribution and labeled according to an unknown target query q_T that we aim to learn. In the *functional* version of this model, a learning algorithm is provided with a sample of examples drawn from the distribution and labeled according to q_T, and must produce a query that agrees with high likelihood with q_T on a random example drawn from the distribution.

More precisely, let P be a probability distribution over data examples and let q_T and q_H be CQs, the target and the hypothesis. The error of q_H relative to q_T and P is

$$\mathsf{error}_{P,q_T,\mathcal{O}}(q_H) = \Pr_{(\mathcal{A},\overline{a})\in P}\big(\mathcal{A}, \mathcal{O} \models q_T(\overline{a}) \text{ and } \mathcal{A}, \mathcal{O} \not\models q_H(\overline{a}), \text{ or } \mathcal{A}, \mathcal{O} \not\models q_T(\overline{a}) \text{ and } \mathcal{A}, \mathcal{O} \models q_H(\overline{a})\big)$$

where $\Pr_{(\mathcal{A},\overline{a})\in P} X$ is the probability of X when drawing $(\mathcal{A}, \overline{a})$ randomly according to P. In other words, $\mathsf{error}_{P,q_T,\mathcal{O}}(q_H)$ is the probability that q_H disagrees with q_T on an example drawn at random from P.

Definition 3.13 (PAC learning algorithm)**.** Let Q be a query class, $\mathcal{L}$ an ontology language. A *PAC learning algorithm* for Q under $\mathcal{L}$ ontologies is a potentially randomized algorithm $\mathbf{A}$ associated with a function $m\colon \mathbb{R}^2 \times \mathbb{N}^4 \to \mathbb{N}$ such that

- $\mathbf{A}$ takes as input a signature Σ, an $\mathcal{L}$ ontology $\mathcal{O}$ with $\mathsf{sig}(\mathcal{O}) \subseteq \Sigma$, and a collection of labeled data examples E;
- for all ϵ, δ with $0 < \epsilon, \delta < 1$, all signatures Σ, all $\mathcal{L}$ ontologies $\mathcal{O}$ with $\mathsf{sig}(\mathcal{O}) \subseteq \Sigma$, all $n_{q_T}, n_{\mathcal{A}} \geq 0$, all probability distributions P over examples $(\mathcal{A}, \overline{a})$ with $\mathsf{sig}(\mathcal{A}) \subseteq \Sigma$ and $\|\mathcal{A}\| \leq n_{\mathcal{A}}$, and all $q_T \in Q_\Sigma$ with $\|q_T\| \leq n_{q_T}$, the following holds:

  When running $\mathbf{A}$ on $\Sigma, \mathcal{O}$, and a collection E of at least $m(\frac{1}{\delta}, \frac{1}{\epsilon}, |\Sigma|, \|\mathcal{O}\|, n_{q_T}, n_{\mathcal{A}})$ labeled data examples drawn from P that are labeled according to q_T under $\mathcal{O}$, $\mathbf{A}$ returns a hypothesis q_H such that with probability at least $1 - \delta$ (over the choice of E), we have $\mathsf{error}_{P,q_T,\mathcal{O}}(q_H) \leq \epsilon$.

We say that $\mathbf{A}$ has *sample size* m and call $\mathbf{A}$ *sample-efficient* if m is a polynomial.

Note that matching Definition 3.10, a PAC learning algorithm is not required to terminate if no fitting query exists.

If there is a (sample-efficient) PAC learning algorithm for Q under $\mathcal{L}$ ontologies, then we say that Q is (*sample-efficiently*) *PAC learnable* under $\mathcal{L}$ ontologies. Furthermore, we say that Q is *polynomial time* PAC learnable under $\mathcal{L}$ ontologies if there is a sample-efficient PAC learning algorithm for Q under $\mathcal{L}$ ontologies that runs in polynomial time in the size of its inputs.

The PAC model according to Definition 3.13 is called *non-uniform* PAC model, in contrast to the *uniform* PAC model, where the sample size may not depend on $\|q_T\|$. It is also known as the *functional* model, in contrast to the *oracle* model. In the oracle model, the algorithm does not receive a collection of labeled examples as input, but has access to an oracle that provides labeled examples upon request. In order for the algorithm to know how many examples it needs to request, it instead receives δ and ϵ as input. Haussler et al. showed that the functional and the oracle model are equivalent, meaning that polynomial time (or sample-efficient) PAC learnability in one model implies polynomial time (or sample-efficient) PAC learnability in the other [Hau+91]. Furthermore, many other small variations of PAC learning also turn out to be equivalent [Hau+91].

For many kinds of logical expressions, polynomial time PAC learnability was found to be impossible, unless some NP-hard problem can be decided in polynomial time. For example, learning 2-term-DNF in polynomial time, would allow solving certain NP-hard graph coloring problems in polynomial time [PV88]. To enable polynomial running time in such cases where PAC learning is hard, PAC

learning is sometimes extended with *membership queries* [AK95], which are defined as in exact learning. Allowing PAC algorithms to make membership queries enables PAC learning 2-term-DNF in polynomial time [Ang88b]. Polynomial PAC learnability also often implies that the corresponding fitting problem can be decided in polynomial time. We discuss this connection further in Chapter 6.

Moreover, there is also a direct connection between exact learning and PAC learning. Angluin showed that polynomial time exact learnability with only equivalence queries implies polynomial time PAC learnability [Ang87; Ang88b]. We show this using our Definitions 3.11 and 3.13 and make the tacit assumption of polynomial time evaluation explicit.

Theorem 3.14 ([Ang88b])**.** *Let $\mathcal{Q}$ be a query class and $\mathcal{L}$ an ontology language, such that $\mathcal{Q}$ is polynomial time learnable under $\mathcal{L}$ ontologies using only equivalence queries. Then $\mathcal{Q}$ is also polynomial time PAC learnable under $\mathcal{L}$ ontologies, given a way to answer $\mathcal{Q}$ queries in polynomial time under $\mathcal{L}$ ontologies.*

Proof. Let $\mathbf{A}$ be an exact learning algorithm for $\mathcal{Q}$ queries under $\mathcal{L}$ ontologies that uses only equivalence queries with running time bound p. Note that p also bounds the number of equivalence queries asked by $\mathbf{A}$. Assume without loss of generality that as soon as the response to an equivalence query is *yes*, $\mathbf{A}$ terminates and returns the hypothesis (and $\mathbf{A}$ does not terminate without receiving a positive response to an equivalence query).

We will use $\mathbf{A}$ to construct a polynomial time PAC learning algorithm $\mathbf{A}'$ for $\mathcal{Q}$ queries under $\mathcal{L}$ ontologies. On input $\mathcal{O}$, Σ and E, $\mathbf{A}'$ first splits E into $n = \lfloor \sqrt{|E|} \rfloor$ segments each of length at least n. It then starts running $\mathbf{A}$ with inputs $\mathcal{O}$ and Σ.

Whenever $\mathbf{A}$ asks an equivalence query with hypothesis q, $\mathbf{A}'$ evaluates q on the labeled examples of the first remaining segment of E, and then discards this segment. If q agrees with all labels in this segment, $\mathbf{A}'$ responds with *yes* to this equivalence query. Otherwise, if there is an example $(\mathcal{A}, \overline{a})$ where q disagrees with the label, $\mathbf{A}'$ returns $(\mathcal{A}, \overline{a})$ to $\mathbf{A}$ as a counterexample and continues running $\mathbf{A}$. If $\mathbf{A}$ terminates with hypothesis q_H, $\mathbf{A}'$ also terminates and returns q_H. If at some point there are no remaining segments, $\mathbf{A}'$ terminates and returns an arbitrary element of $\mathcal{Q}$.

It remains to show that $\mathbf{A}'$ is a polynomial time PAC learning algorithm. Let $\mathcal{O}$ be an $\mathcal{L}$ ontology, Σ a signature, q_T a $\mathcal{Q}$ query and P a probability distribution over examples $(\mathcal{A}, \overline{a})$ with $\|\mathcal{A}\| \leq n_{\mathcal{A}}$. Furthermore, let $p_{\mathbf{A}} = p(|\Sigma|, \|\mathcal{O}\|, \|q_T\|, n_{\mathcal{A}})$ be the running time bound of $\mathbf{A}$ for these inputs, let

$$n = \frac{1}{\epsilon}\left(\ln \frac{1}{\delta} + p_{\mathbf{A}}\right)$$

and let E be a collection of at least n^2 data examples that are labeled according to q_T under $\mathcal{O}$. We have to show that when running $\mathbf{A}'$ on inputs $\mathcal{O}$, Σ and E, it returns a hypothesis q_H such that with at most probability δ, we have $\mathsf{error}_{P,q_T,\mathcal{O}}(q_H) > \epsilon$.

Since $\mathbf{A}$ is guaranteed to identify q_T after at most $p_{\mathbf{A}}$ equivalence queries, $\mathbf{A}'$ must always terminate and return a hypothesis q_H after polynomial time. The probability that $\mathsf{error}_{P,q_T,\mathcal{O}}(q_H) > \epsilon$ is then the probability that $\mathbf{A}'$ responds *yes* to an equivalence query with hypothesis q, although $\mathsf{error}_{P,q_T,\mathcal{O}}(q) > \epsilon$.

The algorithm $\mathbf{A}'$ splits E into at least n segments, each of length at least n. Since $n \geq p_{\mathbf{A}}$, $\mathbf{A}'$ checks each equivalence query of $\mathbf{A}$ against n examples randomly drawn from P. The probability that $\mathbf{A}'$ responds *yes* to an equivalence query with hypothesis q, although $\mathsf{error}_{P,q_T,\mathcal{O}}(q) > \epsilon$, is therefore at most

$$(1-\epsilon)^n \leq e^{-\epsilon n} = \delta \cdot e^{-p_{\mathbf{A}}} \leq \frac{\delta}{p_{\mathbf{A}}}$$

by choice of n. Therefore, the probability that $\mathsf{error}_{P,q_T,\mathcal{O}}(q_H) > \epsilon$ is at most $p_{\mathbf{A}} \cdot \frac{\delta}{p_{\mathbf{A}}} = \delta$, as required. □

Note that the PAC algorithm constructed in this proof makes no use of randomization. Additionally, we can extend this proof to show more expected connections between the two learning models. First, if we extend the PAC learning with membership queries, we can show that polynomial time exact learnability with equivalence and membership queries implies polynomial time PAC learnability with membership queries. Second, if we drop the running time requirement, we can also show that polynomial query learnability with equivalence queries implies sample-efficient PAC learnability (which again can be extended to learning algorithms that additionally have access to membership queries).

Unfortunately, the converse of Theorem 3.14 does not hold. There are concept classes that are polynomial time PAC learnable, but not polynomial time exact learnable with equivalence queries assuming that certain cryptographic problems are hard [Blu94]. Ozaki, Persia, and Mazzullo provide an example of such a concept class in the area of learning description logic ontologies [OPM20].

Chapter 4

Learning with Membership Queries

We begin our investigation into the learnability of queries under ontologies with exact learning algorithms that use only membership queries. Leaving out equivalence queries makes exact learning less powerful, as many query classes that can be learned in polynomial time require both membership and equivalence queries. However, this relatively simpler setting allows us to use a first version of techniques that we in Chapter 5 extend to learning algorithms that also have access to equivalence queries.

From a practical perspective, it is easier to obtain a teacher who answers only membership queries, than a teacher who answers both membership and equivalence queries, as judging whether two queries are equivalent requires more logical expertise. For example, a black-box classifier in place of a query q_T, perhaps a sufficiently large neural network that classifies examples $(\mathcal{A}, \overline{a})$ either positively or negatively, could be used to automatically answer membership queries. Then, a learning algorithm could query this classifier to construct a symbolic representation of q_T. It is, however, unclear how such a classifier could be used to answer equivalence queries.

In this chapter, a learning algorithm receives as input a signature Σ, and an ontology $\mathcal{O}$, and aims to identify a target query q_T by using membership queries.

Example 4.1. Consider the signature $\Sigma = \{A, r\}$, the ontology $\mathcal{O} = \{A \sqsubseteq \exists r.\top\}$ and that we aim to learn ELQs. If the target ELQ is

$$q_T(x_0) \leftarrow r(x_0, x_1) \wedge A(x_1),$$

then a learner can identify q_T (up to $\equiv_{\mathcal{O}}$-equivalence) using membership queries as follows. First, by asking whether $\{r(a_1, a_2), A(a_2)\}, \mathcal{O} \vDash q_T(a_1)$ and receiving *yes* as a response, the learner can eliminate ELQs such as $q_1(x_0) \leftarrow A(x_0)$ and $q_2(x_0) \leftarrow r(x_0, x_1) \wedge r(x_1, x_2) \wedge r(x_1, x_2)$ as well as anything of greater size as possibilities for q_T. Then, by asking whether $\{r(b_1, b_2), r(b_2, b_3)\}, \mathcal{O} \vDash q_T(b_1)$ and receiving *no* as a response, the learner can eliminate ELQs such as $q_3(x_0) \leftarrow r(x_0, x_1) \wedge r(x_1, x_2)$ and $q_4(x_0) \leftarrow r(x_0, x_1)$ as candidates. The only ELQ that agrees with these answers is q_T.

The first observation we make about this setting of only membership queries is that the property of being (polynomial time) learnable using only membership queries is anti-monotone in both the query class $\mathcal{Q}$ and the ontology language $\mathcal{L}$. This is not automatically the case when equivalence queries are used. Assume that $\mathcal{Q}$ is (polynomial time) learnable under $\mathcal{L}$ ontologies. Further, let $\mathcal{Q}'$ be a query class and $\mathcal{L}'$ be an ontology language such that $\mathcal{Q}' \subseteq \mathcal{Q}$ and $\mathcal{L}' \subseteq \mathcal{L}$. Then $\mathcal{Q}'$ is also (polynomial time) learnable under $\mathcal{L}'$ ontologies[1].

The second observation is that there is an interesting necessary precondition for a query class to be learnable with only membership queries, namely the existence of finite *unique characterizations*.

Definition 4.2 (Unique characterization)**.** Let $\mathcal{Q}$ be a query class, $\mathcal{O}$ an ontology and $q \in \mathcal{Q}$. A set of labeled examples E is a *unique characterization* of q under $\mathcal{O}$, if q fits E under $\mathcal{O}$ and $q \equiv_{\mathcal{O}} p$ for all $p \in \mathcal{Q}$ that fit E under $\mathcal{O}$.

A query $q \in \mathcal{Q}$ is *uniquely characterizable* (with regard to $\mathcal{Q}$) if there exists a unique characterization of q.

Example 4.3. Consider $\mathcal{O}$ as in Example 4.1. The ELQ $q(x_0) \leftarrow r(x_0, x_1) \wedge A(x_1)$ is uniquely characterized (with regard to all ELQs over the signature $\{r, A\}$) under $\mathcal{O}$ by the positive data example $(\{r(a_1, a_2), A(a_2)\}, a_2)$ and the negative data example $(\{r(b_1, b_2), r(b_2, b_3)\}, b_1)$.

Unique characterizations are closely connected to learning with only membership queries. Let us assume that there is a learning algorithm $\mathbf{A}$ for a query class $\mathcal{Q}$ under $\mathcal{L}$ ontologies that uses only membership queries. When $\mathbf{A}$ is started on a signature Σ as well as an ontology $\mathcal{O}$ and then identifies a target query q_T, it must have done so based on the answers to its membership queries. Since $\mathbf{A}$ must be able to identify all queries in $\mathcal{Q}$ under all $\mathcal{L}$ ontologies, the data examples used in the membership queries combined with the answers provided by the teacher must form a finite unique characterization of q_T under $\mathcal{O}$. If $\mathbf{A}$ is a polynomial query learning algorithm, then the resulting unique characterization must even be of polynomial size in $\|q_T\|$, $\|\mathcal{O}\|$ and $|\Sigma|$.

Moreover, an algorithm that computes finite unique characterizations of $\mathcal{Q}$ queries allows us to construct a learning algorithm for $\mathcal{Q}$ queries that uses only membership queries. The learning algorithm enumerates all $q \in \mathcal{Q}$ ordered by size, and for each query q in the enumeration it computes a unique characterization E. It then uses membership queries to check whether the target query q_T fits E. If this is the case, then $q \equiv_{\mathcal{O}} q_T$ and the learning algorithm returns q. Otherwise, it continues with the next query. Unfortunately, this algorithm always requires at least an

[1] With the additional requirement that every $q \in \mathcal{Q}$ that is equivalent to a query in $\mathcal{Q}'$ can be transformed into a $\mathcal{Q}'$ query in polynomial time.

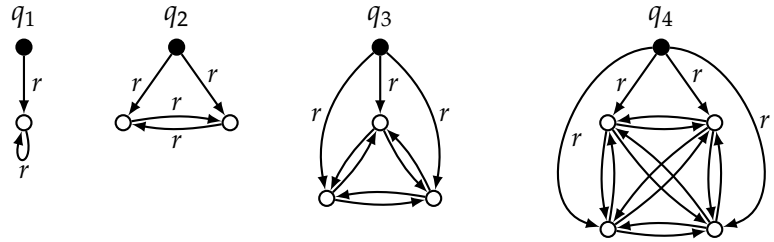

Figure 4.1: CQs that are not uniquely characterizable.

exponential number of membership queries for our query classes even if all unique characterizations are of polynomial size, so polynomial time learnability cannot be obtained this way. Our learning algorithm will use a smarter strategy.

Ten Cate, Dalmau, and Kolaitis show that the class of all CQs is not polynomial query learnable using only membership queries, even under the empty ontology [tCDK13]. They use an argument in the style of Lemma 3.12 to show that even if the size of a target CQ is known to the learning algorithm in advance, at least an exponential number of membership queries is necessary to identify it. To understand why learning CQs is not possible with a polynomial number of membership queries, consider for every $k \geq 1$ the CQ

$$q_k(x_0) \leftarrow \bigwedge_{1 \leq i \leq k} r(x_0, x_i) \wedge \bigwedge_{\substack{1 \leq i,j \leq k \\ i \neq j}} r(x_i, x_j).$$

The queries q_1, q_2, q_3 and q_4 are displayed in Figure 4.1, where the unlabeled edges are also r-atoms. We can see that every unique characterization of q_1 must be infinite. If there were a finite unique characterization E of q, then there must be a number n that is larger than the length of the longest cycle in every negative example in E. Consider then the query q_n. Either, q_n correctly labels all negative examples in E, or there is a negative example $(\mathcal{B}, b)$ that is labeled positively by q_n. In the first case, since $q_1 \subseteq_\varnothing q_n$, q_n fits all examples in E, contradicting that E is a unique characterization. In the second case, since $n > \mathsf{ind}(\mathcal{B})$, it must also be the case that q_1 labels $(\mathcal{B}, b)$ positively, contradicting that q_1 fits E.

The cycles in every q_k are crucial for this argument. Indeed, the class of c-acyclic CQs, that heavily restricts the occurrence of cycles, is uniquely characterizable. Most importantly, it can also be shown that c-acyclic CQs are polynomial time learnable using only membership queries [tCD22]. This class of CQs contains all ELIQs, which implies the following.

Proposition 4.4 ([tCD22])**.** *ELIQs are polynomial time learnable under the empty ontology using only membership queries.*

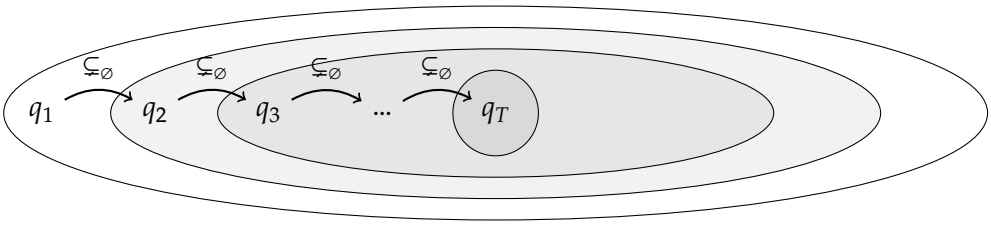

Figure 4.2: The sequence $q_1, q_2, \ldots$ approximates q_T from above. Initially, all queries are candidates for q_T, as $q_1 \subseteq_\varnothing p$ holds for all p. Each move from q_i to q_{i+1} eliminates some candidates for q_T.

In our investigation into the learnability of queries under ontologies using only membership queries, we therefore focus on ELIQs a large class of queries that is potentially learnable in polynomial time.

We also base our approach on the learning algorithm behind Proposition 4.4, whose strategy is visualized in Figure 4.2. This algorithm identifies q_T in polynomial time by starting with a very specific query q_1 such that it is guaranteed that $q_1 \subseteq_\varnothing q_T$, and then constructing with the help of membership queries a sequence of increasingly more general hypotheses $q_1, q_2, \ldots$, where in each step the number of queries p such that $q_i \subseteq_\varnothing p$ is reduced, but the property that $q_i \subseteq_\varnothing q_T$ is maintained. This sequence must then arrive at q_T after polynomially many steps.

This approach heavily relies on the fact that for classes of CQs, membership queries can be used to imitate subset queries due to Lemma 3.7. A learning algorithm can check whether $q_i \subseteq_\varnothing q_T$ by asking the teacher whether $\mathcal{A}_{q_i}, \varnothing \models q_T(\overline{x}_i)$ using a membership query.

In this chapter, we generalize the central notions of this algorithm to the case with ontologies in order to establish learnability results for the class of all ELIQs.

Structure of this Chapter

First, in Section 4.1, we look into the limits of learning ELIQs under ontologies and identify several ontology languages under which polynomial query learning is impossible. This motivates a restriction of functionality constraints in $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}}$ ontologies, resulting in the ontology language $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}-}$.

Then, we extend the techniques used to show Proposition 4.4 to $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}-}$. In Section 4.2, we show that we can restrict our attention to ontologies in normal form: if a class of queries is polynomial time learnable under ontologies in normal form, then it is also polynomial time learnable under unrestricted ontologies.

In Section 4.3, we look at the core operation of the learning algorithm: moving from q_i to q_{i+1} while maintaining $q_i \subseteq_{\mathcal{O}} q_T$. For this, the algorithm needs to construct

the set of all *most specific* generalizations of q_i, called a *frontier* of q_i. We show that ELIQs always possess frontiers of polynomial size under *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ ontologies, and we can search them in polynomial time to find a suitable candidate for q_{i+1}. Additionally, we show that many extensions of *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ do not permit frontiers of polynomial size.

Then, in Section 4.4, we show that even under ontologies, the sequence $q_1, q_2, \ldots$ must reach q_T after at most a polynomial number of steps.

What remains is to find a suitable q_1 for the start of this sequence, which is no longer trivial under ontologies. In Section 4.5 we determine how a learning algorithm can obtain it. In the worst case, if the ontology contains unrestricted concept disjointness constraints, then we show that a single equivalence query cannot be avoided to obtain q_1.

We put the results of the previous sections together in Section 4.6 to present the complete learning algorithm for ELIQs under *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ and show that it runs in polynomial time. This is the main result of this chapter.

Finally, we end this chapter with a discussion about these results in Section 4.7

Related Publications

Much of the material in this chapter is based on the publications [FJL21b; FJL22a; FJL22b]. However, many of the results have been generalized. The main results, Theorem 4.23 about the existence of frontiers and Theorem 4.42 about the polynomial time learnability of ELIQs, were previously only shown to hold for *DL-Lite*$_{\text{core}}^{\mathcal{H}}$ and *DL-Lite*$_{\text{core}}^{\mathcal{F}^-}$, but are shown here to hold for *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$.

4.1 Limits of Membership Queries

We know that ELIQs are polynomial time learnable under the empty ontology using only membership queries, but CQs are not. Under which ontology sublanguages of $\mathcal{ELIHF}_\bot$ can we hope to learn queries in polynomial time using only membership queries? Unfortunately, already for $\mathcal{EL}$ or *DL-Lite*$_{\text{horn}}$ ontologies, an exponential number of membership queries can be necessary to identify the target query. The perhaps surprising reason for this is that these ontology languages can express conjunctions like $A_1 \sqcap A_2 \sqsubseteq B$. We follow a strategy similar to the proof of Lemma 3.12 in order to show that using simple inclusions of conjunctions of concept names suffices to make polynomial query learning impossible, even for queries that only consist of conjunctions of concept name atoms.

A *conjunction of atomic queries* is a unary CQ of the form $q(x_0) \leftarrow A_1(x_0) \wedge \cdots \wedge A_n(x_0)$ where $A_1, \ldots, A_n$ are concept names. Every query that is a conjunction of atomic

queries is also an ELIQ and an ELQ.

Theorem 4.5. *Conjunctions of atomic queries are not polynomially query learnable under conjunctive ontologies using only membership queries*

Proof. For each $n \geq 1$, we use the set

$$S_n = \{q(x) \leftarrow \alpha_1(x) \wedge \ldots \wedge \alpha_n(x) \mid \alpha_i \in \{A_i, B_i\} \text{ for all } i \text{ with } 1 \leq i \leq n\},$$

of 2^n conjunctions of atomic queries that use the signature $\Sigma = \{A_i, B_i \mid 1 \leq i \leq n\}$, and we use the conjunctive ontology

$$\mathcal{O}_n = \{A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B'_n \mid 1 \leq i \leq n\}.$$

Assume to the contrary of what is to be shown that conjunctions of atomic queries are polynomial query learnable under conjunctive ontologies. Then there exists a learning algorithm and polynomial p such that the number of membership queries is bounded by $p(n_\Sigma, n_\mathcal{O}, n_{q_T})$, where n_Σ is the size of the signature Σ, $n_\mathcal{O}$ the size of the ontology and n_{q_T} the size of the target query[2]. We choose n such that

$$2^n > p(2n, \|\mathcal{O}_n\|, r(n)),$$

where r is a polynomial such that every query $q \in S_m$ satisfies $\|q\| = r(m)$ for every $m \geq 1$.

Now, consider a membership query posed by the learning algorithm with the data example $(\mathcal{A}, a)$. The teacher responds as follows:

1. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for no $q \in S_n$, then answer *no*
2. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for a single $q \in S_n$, then answer *no* and remove q from S_n
3. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for more than one $q \in S_n$, then answer *yes*.

Note that the third response is consistent since $\mathcal{A}$ must then contain $A_i(a)$ and $B_i(a)$ for some i and thus $\mathcal{O}_n$ implies that a is an answer to every query in S_n. Moreover, the answers are always correct with respect to the updated set S_n. Thus, the learner cannot distinguish the remaining candidate queries by answers to queries posed so far.

It follows that the learning algorithm removes at most $p(2n, \|\mathcal{O}_n\|, r(n))$ queries from S_n. By the choice of n, at least two candidates remain in S_n after the algorithm asks the last membership query. Thus, the learner cannot distinguish between them, and we have derived a contradiction. □

[2] As no equivalence queries are allowed, the size $n_\mathcal{A}$ of the largest counterexample is fixed.

Recall that the long concept inclusions of the form $A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B'_n$ can be expressed through several simpler concept inclusions $A_i \sqcap B_i \sqsubseteq A_1$, $A_i \sqcap B_i \sqsubseteq B_1$, ..., $A_i \sqcap B_i \sqsubseteq B'_n$. Hence, only conjunctions on the left side of concept inclusions are necessary for the ontology used in the proof of Theorem 4.5.

Therefore, only DLs of the *DL-Lite*$_{\text{core}}$ family are candidates for polynomial time learning of queries, since they restrict the use of conjunctions on the left side of concept inclusions. Next, we show that ELIQs are *not learnable* at all using only membership queries, if functionality constraints and existential restrictions interact in an ontology.

Theorem 4.6. *ELIQs are not learnable under DL-Lite*$^{\mathcal{F}}_{\text{core}}$ *ontologies using only membership queries*

Proof. The proof follows a structure similar to the proof of Theorem 4.5, but as we aim to show non-learnability instead of non-polynomial query learnability, we will use an infinite set S of candidate target queries.

We use the fixed *DL-Lite*$^{\mathcal{F}}_{\text{core}}$ ontology

$$\mathcal{O} = \{\, A \sqsubseteq \exists r.\top, \quad \exists r^-.\top \sqsubseteq \exists r.\top, \quad \exists r.\top \sqsubseteq \exists s.\top, \quad \mathsf{func}(r^-) \,\}.$$

and the set

$$S = \{q^*\} \cup \{q_n \mid n \text{ is prime}\},$$

where $q^*(x_1) \leftarrow A(x_0) \wedge r(x_0, x_1) \wedge A(x_1)$ and each q_n is defined as follows:

$$\begin{aligned} q_n(x_1) \leftarrow & A(x_0) \wedge r(x_0, x_1) \wedge r(x_1, x_2) \wedge \cdots \wedge r(x_{n-1}, x_n) \wedge \\ & s(x_n, y) \wedge s(x'_n, y) \wedge \\ & r(x'_1, x'_2) \wedge \cdots \wedge r(x'_{n-1}, x'_n) \wedge A(x'_1). \end{aligned}$$

The queries q^* and q_n are displayed in Figure 4.3. It is important to note that $q^* \subseteq_{\mathcal{O}} q_n$ for all $n \geq 1$ and $q_i \not\subseteq_{\mathcal{O}} q_j$ for all $i \neq j$. Intuitively, this makes it impossible for the learner to distinguish between q^* being the target query and one of the q_n being the target query. If, for example, the learner asks a membership query with the data example $(\mathcal{A}_{q^*}, x_1)$ then the teacher will answer *yes*, and the learner has not gained any additional information. The following claim describes the main property of the chosen queries.

Claim. Let $\mathcal{A}$ be an ABox and $a \in \mathsf{ind}(\mathcal{A})$. If $\mathcal{A}, \mathcal{O} \not\models q^*(a)$, then $\mathcal{A}, \mathcal{O} \models q_n(a)$ for only finitely many primes n.

Proof of the claim. Let $\mathcal{A}$ be an ABox and $a \in \mathsf{ind}(\mathcal{A})$ such that $\mathcal{A}, \mathcal{O} \not\models q^*(a)$. Then $\mathcal{A}$ must satisfy $\mathsf{func}(r^-)$. Suppose to the contrary of what we have to show that there

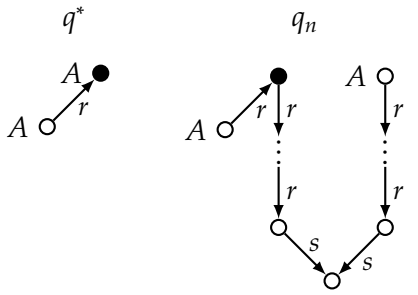

Figure 4.3: The queries q^* and q_n.

are infinitely many primes n such that $\mathcal{A}, \mathcal{O} \models q_n(a)$ and let h_n be the witnessing homomorphisms from q_n to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ with $h_n(x_1) = a$. We distinguish cases.

If $h_n(x_n) \notin \mathsf{ind}(\mathcal{A})$ for some prime n with $\mathcal{A}, \mathcal{O} \models q_n(a)$, then $h_n(x'_n) = h_n(x_n)$ due to the tree structure of the traces in $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. Since r^- is functional in $\mathcal{U}_{\mathcal{A},\mathcal{O}}$, it follows that $h_n(x_j) = h_n(x'_j)$ for all j with $1 \leq j \leq n$. Since $A(x'_1) \in q_n$, also $A(h_n(x'_1)) = A(h_n(x_1)) = A(a) \in \mathcal{A}$. Since also $A(x_0), r(x_0, x_1) \in q_n$, we have $h_n(x_0) \in A^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$, $(h_n(x_0), h_n(x_1)) \in r^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$, and thus h_n is a homomorphism from q^* to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ with $h_n(x_1) = a$. Hence, $\mathcal{A}, \mathcal{O} \models q^*(a)$, a contradiction.

Otherwise, $h_n(x_n) \in \mathsf{ind}(\mathcal{A})$ for all primes n with $\mathcal{A}, \mathcal{O} \models q_n(a)$. Since $\mathcal{A}$ is finite, there is an element $b \in \mathsf{ind}(\mathcal{A})$ such that $h_m(x_m) = b$ for infinitely many primes m. Thus, there is an r-path of length m from a to b in $\mathcal{A}$ for infinitely many primes m. Since $\mathcal{A}$ is finite and satisfies $\mathsf{func}(r^-)$, this is only possible if $a = b$, $r(a, a) \in \mathcal{A}$, and $h_m(x_j) = a$ for all considered m and all j with $1 \leq j \leq m$. Since also $A(x_0), r(x_0, x_1) \in q_n$ and $\mathcal{A}$ satisfies $\mathsf{func}(r^-)$, we further have $h_n(x_0) = a$ for all primes n with $\mathcal{A}, \mathcal{O} \models q_n(a)$ and $A(a) \in \mathcal{A}$. But then $\mathcal{A}, \mathcal{O} \models q^*(a)$, a contradiction.

Now, assume there is a learning algorithm for ELIQs under $\mathit{DL\text{-}Lite}^{\mathcal{F}}_{\text{core}}$ ontologies that uses only membership queries, and consider a membership query posed by this algorithm with the data example $(\mathcal{A}, a)$. An adversarial teacher can respond as follows:

1. if $\mathcal{A}, \mathcal{O} \models q^*(a)$, then reply *yes*;
2. otherwise, reply *no* and remove from S every q_n that satisfies $\mathcal{A}, \mathcal{O} \models q_n(a)$.

An important aspect of this strategy is that, as proved in the claim, only finitely many hypotheses q are removed whenever Case 2 above applies. Consequently, after any number of membership queries, the set of remaining hypotheses S is infinite and contains q^*. The learning algorithm then can, however, not distinguish between q^* and the remaining hypotheses and thus not identify the target query. In

particular, the presence of $q^* \in S$ prevents the learning from simply going through all $q_i \in S$, asking membership queries with ABoxes that take the form of these queries, and identifying q_i as the target query when the membership query for q_i succeeds. □

Note that this is a strong result: no amount of membership queries suffices to identify q^*, which also means that no finite unique characterization of q^* under $\mathcal{O}$ exists. Hence, in order to learn ELIQs under ontologies that feature functionality constraints, we have to restrict the interaction between existential restrictions and functionality constraints.

A *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontology $\mathcal{O}$ is a *DL-Lite*$^{\mathcal{HF}}_{\text{core}}$ ontology where for all $\mathsf{func}(r) \in \mathcal{O}$, the existential restriction $\exists r^-.\top$ does not occur on the right side of a concept inclusion in $\mathcal{O}$. This excluded the ontology used in the proof of Figure 4.3. We focus on learning ELIQs under *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontologies in the following sections of this chapter. *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ is, according to the results in this section, in some sense a maximal ontology language, for which polynomial time learning with only membership queries is possible.

4.2 Reducing to Ontologies in Normal Form

When working with DL ontologies, it is often useful to assume that they are in *normal form*. This allows a simpler presentation of algorithms and proofs. In particular, we assume in the following sections and chapters that the ontology that a learning algorithm receives as input is in normal form. In this section, we show that this assumption is not essential. Any learning algorithm that learns queries under ontologies in normal form can be converted into a learning algorithm for queries under unrestricted ontologies, without changing the number of membership or equivalence queries the algorithm performs and only requiring a polynomial amount of additional time. The idea behind this is that we can rewrite ontologies into normal form in a way that enables us to translate membership queries and equivalence queries for the teacher, who still answers questions under the original ontology. Although we only need to consider *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontologies in this chapter, we show this result for all $\mathcal{ELIHF}_\bot$ ontologies, which we will need in the next chapter.

An $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}'$ is a *conservative extension* of an $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}$ if

- $\mathsf{sig}(\mathcal{O}) \subseteq \mathsf{sig}(\mathcal{O}')$,
- every model of $\mathcal{O}'$ is a model of $\mathcal{O}$, and

- for every model $\mathcal{I}$ of $\mathcal{O}$ there exists a model $\mathcal{I}'$ of $\mathcal{O}'$ such that $S^{\mathcal{I}} = S^{\mathcal{I}'}$ for all concept or role names $S \notin \mathsf{sig}(\mathcal{O}') \setminus \mathsf{sig}(\mathcal{O})$.

It is well known that every $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}$ can be transformed in polynomial time into an $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}'$ in normal form by introducing new concept names such that $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$ [Baa+17].

For the purpose of learning algorithms, the properties of conservative extensions are not sufficient. We require a specific method to translate queries and examples from a signature that contains newly introduced concept names to a signature that uses only symbols from Σ. Hence, we require a specific rewriting of ontologies into normal form.

Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology and Σ a signature with $\mathsf{sig}(\mathcal{O}) \subseteq \Sigma$. We construct an $\mathcal{ELIHF}_\bot$ ontology $\mathcal{O}'$ in normal form as follows. Let $\mathsf{sub}(\mathcal{O})$ denote the set of all concepts that occur in concept inclusions in $\mathcal{O}$, that is,

$$\mathsf{sub}(\mathcal{O}) = \bigcup_{C \sqsubseteq D \in \mathcal{O}} \mathsf{sub}(C) \cup \mathsf{sub}(D)$$

where

$$\begin{aligned}
\mathsf{sub}(\top) &= \{\top\} \\
\mathsf{sub}(A) &= \{A\} \\
\mathsf{sub}(C \sqcap D) &= \{C \sqcap D\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D) \\
\mathsf{sub}(\exists R.C) &= \{\exists R.C\} \cup \mathsf{sub}(C).
\end{aligned}$$

With each concept $C \in \mathsf{sub}(\mathcal{O})$, we associate a concept name X_C. If $C = \top$, set $X_C = \top$, if C is a concept name A, set $X_A = A$. Otherwise, let X_C be a fresh concept name not contained in Σ. We use $\mathbf{X}$ to refer to the set of all of these fresh concept names.

The new ontology $\mathcal{O}'$ consists of all role inclusions, functionality constraints, role disjointness constraints, and concept disjointness constraints in $\mathcal{O}$ as well as the following concept inclusions:

- $X_C \sqsubseteq X_D$ for every $C \sqsubseteq D \in \mathcal{O}$,
- $X_{D_1 \sqcap D_2} \sqsubseteq X_{D_i}$ and $X_{D_1} \sqcap X_{D_2} \sqsubseteq X_{D_1 \sqcap D_2}$, for every $D_1 \sqcap D_2 \in \mathsf{sub}(\mathcal{O})$ and $i \in \{1, 2\}$,
- $X_{\exists R.C} \sqsubseteq \exists R.X_C$ and $\exists R.X_C \sqsubseteq X_{\exists R.C}$, for every $\exists R.C \in \mathsf{sub}(\mathcal{O})$.

Since $|\mathsf{sub}(\mathcal{O})|$ is polynomial in $\|\mathcal{O}\|$, $\mathcal{O}'$ can be computed in polynomial time.

Example 4.7. For $\mathcal{O} = \{\exists r.(A \sqcap B) \sqsubseteq A \sqcap B\}$, the set $\mathsf{sub}(\mathcal{O})$ is $\{A, B, A \sqcap B, \exists r.(A \sqcap B)\}$. Hence, the above construction produces

$$\begin{aligned}\mathcal{O}' = \{&X_{\exists r.(A\sqcap B)} \sqsubseteq X_{A\sqcap B},\\ &X_{A\sqcap B} \sqsubseteq A,\ X_{A\sqcap B} \sqsubseteq B,\ A \sqcap B \sqsubseteq X_{A\sqcap B},\\ &X_{\exists r.(A\sqcap B)} \sqsubseteq \exists r.X_{A\sqcap B},\ \exists r.X_{A\sqcap B} \sqsubseteq X_{\exists r.(A\sqcap B)}\},\end{aligned}$$

which is in normal form.

We observe the following consequences of the construction of $\mathcal{O}'$ regarding the relationship between $\mathcal{O}$ and $\mathcal{O}'$.

Lemma 4.8. *Let $\mathcal{L}$ be a ontology language contained in $\mathcal{ELIHF}_\bot$ and $\mathcal{O}$ an $\mathcal{L}$ ontology. Then,*

1. *$\mathcal{O}'$ is an $\mathcal{L}$ ontology in normal form;*
2. *$\mathcal{O}'$ is a conservative extension of $\mathcal{O}$;*
3. *$\mathsf{sig}(\mathcal{O}') = \mathsf{sig}(\mathcal{O}) \cup \mathbf{X}$;*
4. *$\mathcal{O}' \models X_C \equiv C$ for all $C \in \mathsf{sub}(\mathcal{O})$.*

Lemma 4.8 tells us that $\mathcal{O}'$ is not only a conservative extension of $\mathcal{O}$, but $\mathcal{O}'$ also specifies how precisely a model of $\mathcal{O}$ can be extended to a model of $\mathcal{O}'$. This specific rewriting into normal form allows us to translate data examples and queries from the signature $\Sigma \cup \mathbf{X}$ to the signature Σ. We describe this translation next.

Let $\mathcal{A}$ be an ABox with $\mathsf{sig}(\mathcal{A}) \subseteq \Sigma \cup \mathbf{X}$. Then, $\mathcal{A}_\Sigma$ with $\mathsf{sig}(\mathcal{A}_\Sigma) \subseteq \Sigma$ is obtained by starting with $\mathcal{A}$ and exhaustively applying the following operation.

Replace concept name Choose an assertion $X_C(a) \in \mathcal{A}_\Sigma$ with $X_C \in \mathbf{X}$ and remove it.

- If $C = D_1 \sqcap D_2$ for some concepts D_1 and D_2, add the assertions $X_{D_1}(a)$ and $X_{D_2}(a)$.
- If $C = \exists R.D$ for some concept D and role R, then if $\mathsf{func}(R) \in \mathcal{O}$ and there is an assertion $R(a, b) \in \mathcal{A}_\Sigma$, add the assertion $X_D(b)$, otherwise introduce a fresh individual name a' and add the assertion $R(a, a')$ and $X_D(a')$.

When *Replace concept name* is no longer applicable, then $\mathcal{A}_\Sigma$ uses no symbols from $\mathbf{X}$. This is the case after a polynomial number of applications of *Replace concept name*, as

$$\sum_{\substack{X_C(a)\in\mathcal{A}_\Sigma\\ X_C\in\mathbf{X}}} \|C\|$$

decreases with each application.

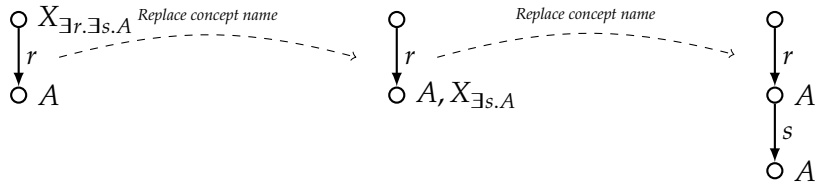

Figure 4.4: An application of *Replace concept name* to create $\mathcal{A}_\Sigma$ with $\mathsf{func}(r) \in \mathcal{O}$.

Example 4.9. Consider the ontology $\mathcal{O} = \{B \sqsubseteq \exists r.\exists s.A,\ \mathsf{func}(r)\}$. Figure 4.4 shows an example of two applications of *Replace concept name* to remove the introduced concept name $X_{\exists r.\exists s.A}$.

As CQs can be viewed as ABoxes, we extend this construction to CQs and write q_Σ for $\mathcal{A}_{q_\Sigma}$ viewed as a query. Using the definition of *Replace concept name*, it then can be verified, that this construction preserves the query class of q in the following sense.

Lemma 4.10. *If $\mathcal{Q} \in \{\mathrm{CQ}, \mathrm{ELIQ}\}$ and $q \in \mathcal{Q}$, then $q_\Sigma \in \mathcal{Q}$. If q is an ELQ and $\mathcal{O}$ is an $\mathcal{EL}^r$ ontology, then q_Σ is also an ELQ.*

Next, we show that this translation has the properties we need to preserve the answers to membership and equivalence queries. Let $\mathcal{B}$ be an ABox. With $\mathcal{B}|_\Sigma$ we denote the restriction of $\mathcal{B}$ to symbols in Σ, that is,

$$\mathcal{B}|_\Sigma = \{A(a) \in \mathcal{B} \mid A \in \Sigma\} \cup \{r(a,b) \in \mathcal{B} \mid r \in \Sigma\}.$$

The following lemma captures the properties of the construction of $\mathcal{A}_\Sigma$ and q_Σ.

Lemma 4.11. *The following holds:*

1. *For all k-ary CQs q that only use symbols from Σ, all ABoxes $\mathcal{A}$ and all $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$, $\mathcal{A}, \mathcal{O}' \models q(\overline{a})$ if and only if $\mathcal{A}_\Sigma, \mathcal{O} \models q(\overline{a})$.*
2. *For all k-ary CQs q that use only symbols from $\Sigma \cup \mathbf{X}$, all ABoxes $\mathcal{B}$ and all $\overline{a} \in \mathsf{ind}(\mathcal{B})^k$, $\mathcal{B}|_\Sigma, \mathcal{O}' \models q(\overline{a})$ if and only if $\mathcal{B}, \mathcal{O} \models q_\Sigma(\overline{a})$.*

Proof. We begin by showing Point 1. For the *if* direction, suppose that $\mathcal{A}_\Sigma, \mathcal{O} \models q(\overline{a})$ and let $\mathcal{I}$ be a model of $\mathcal{A}$ and $\mathcal{O}'$. We can assume that $\mathcal{I}$ does not mention any of the individuals that were introduced in the construction of $\mathcal{A}_\Sigma$ by the *Replace concept name* operation. We will extend $\mathcal{I}$ to a model $\mathcal{I}_\Sigma$ of $\mathcal{A}_\Sigma$ and $\mathcal{O}$, such that $\mathcal{I}_\Sigma, \overline{a} \to \mathcal{I}, \overline{a}$. This suffices since $\mathcal{I}_\Sigma \models q(\overline{a})$. We cannot use $\mathcal{I}$ directly as a model of $\mathcal{A}_\Sigma$ as it may not interpret the fresh individual names in $\mathcal{A}_\Sigma$.

We construct $\mathcal{I}_\Sigma$ by processing every assertion introduced by the *Replace concept name* operation. Let $R(b, b')$ be such an assertion. Then, $X_{\exists R.C}(b) \in \mathcal{A}$ and, by Lemma 4.8, $\mathcal{O}' \models X_{\exists R.C} \sqsubseteq \exists R.X_C$. Since $\mathcal{I}$ is a model of $\mathcal{A}$ and $\mathcal{O}'$, there is an element c with $R(b, c) \in \mathcal{I}$. Informally, let $\mathcal{J}_c$ be the *unraveling* of $\mathcal{I}$ at c which takes into account the functionality constraints in $\mathcal{O}$, and in which the R^--successor of c is omitted in case $\mathsf{func}(R^-) \in \mathcal{O}$. Then, add a copy of $\mathcal{J}_c$ to $\mathcal{I}$, rename the root of $\mathcal{J}_c$ to b', and add (b, b') to $R^\mathcal{I}$.

We now give a formal definition of $\mathcal{J}_c$. Its domain $\Delta^{\mathcal{J}_c}$ consists of all sequences $a_0 R_1 a_1 \cdots R_n a_n$ such that

- $a_0 = c$;
- $a_i \in \Delta^\mathcal{I}$, for all i with $0 \leq i \leq n$;
- $(a_i, a_{i+1}) \in R_{i+1}^\mathcal{I}$, for all i with $0 \leq i < n$;
- if $\mathsf{func}(R_i^-) \in \mathcal{O}$, then $R_{i+1} \neq R_i^-$, for all i with $0 \leq i < n$;
- if $R_1 = R^-$ then $\mathsf{func}(R^-) \notin \mathcal{O}$.

The interpretation of concept and role names is then as expected:

$$\begin{aligned} A^{\mathcal{J}_c} &= \{a_0 R_1 a_1 \cdots R_n a_n \in \Delta^{\mathcal{J}_c} \mid a_n \in A^\mathcal{I}\} && \text{for all } A \in \mathsf{N_C}; \\ r^{\mathcal{J}_c} &= \{(p, pra) \mid pra \in \Delta^{\mathcal{J}_c}\} \cup \\ &\quad \{(pr^- a, p) \mid pr^- a \in \Delta^{\mathcal{J}_c}\} && \text{for all } r \in \mathsf{N_R}. \end{aligned}$$

Note that the function that maps every sequence $a_0 R_1 \ldots a_n$ to a_n is a homomorphism from $\mathcal{J}_c$ to $\mathcal{I}$.

Let $\mathcal{I}_\Sigma$ be the result of doing the above for every atom in $\mathcal{A}_\Sigma \setminus \mathcal{A}$. By construction of $\mathcal{I}_\Sigma$, we know that $\mathcal{I}_\Sigma$ is a model of $\mathcal{A}_\Sigma$. It is routine to verify that $\mathcal{I}_\Sigma$ is also a model of $\mathcal{O}$ and that there is a homomorphism h from $\mathcal{I}_\Sigma$ to $\mathcal{I}$ with $h(\overline{a}) = \overline{a}$.

For the *only if* direction of Point 1, suppose that $\mathcal{A}, \mathcal{O}' \models q(\overline{a})$ and let $\mathcal{I}$ be a model of $\mathcal{A}_\Sigma$ and $\mathcal{O}$. Since $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$, there is a model $\mathcal{I}'$ of $\mathcal{O}'$ that coincides with $\mathcal{I}$ on $\mathsf{sig}(\mathcal{O})$. Moreover, by Point 4 of Lemma 4.8, it is also a model of $\mathcal{A}$. It follows that $\mathcal{I} \models q(\overline{a})$ as required.

For the *if* direction of Point 2, suppose that $\mathcal{B}, \mathcal{O} \models q_\Sigma(\overline{a})$ and let $\mathcal{I}$ be a model of $\mathcal{B}|_\Sigma$ and $\mathcal{O}'$. Since q_Σ contains only symbols from Σ, $\mathcal{B}|_\Sigma, \mathcal{O} \models q_\Sigma(\overline{a})$. Since $\mathcal{O}'$ is a conservative extension of $\mathcal{O}$, $\mathcal{I}$ is also a model of $\mathcal{O}$. Thus, $\mathcal{I} \models q_\Sigma(\overline{a})$ and by Point 4 of Lemma 4.8, $\mathcal{I} \models q(\overline{a})$ follows as required.

For the *only if* direction of Point 2, suppose $\mathcal{B}|_\Sigma, \mathcal{O}' \models q(\overline{a})$ and let $\mathcal{I}$ be a model of $\mathcal{A}$ and $\mathcal{O}$. Then the restriction $\mathcal{I}|_\Sigma$ of $\mathcal{I}$ to symbols in Σ is a model of $\mathcal{B}|_\Sigma$ and by $\mathcal{O}'$ being a conservative extension of $\mathcal{O}$, there is a model $\mathcal{I}'$ of $\mathcal{O}'$ that coincides with

$\mathcal{I}|_\Sigma$ on all symbols from Σ. Thus, $\mathcal{I}' \models q(\overline{a})$ and since $\mathcal{I}'$ is a model of $\mathcal{O}'$, $\mathcal{I}' \models q_\Sigma(\overline{a})$. It follows that $\mathcal{I}|_\Sigma \models q_\Sigma(\overline{a})$. As q_Σ uses only symbols from Σ, $\mathcal{I} \models q_\Sigma(\overline{a})$ follows, as required. □

With these properties of $\mathcal{A}_\Sigma$ and q_Σ in hand, we show the main result of this section: polynomial time learnability under ontologies in normal form implies polynomial time learnability under unrestricted ontologies. As we will use this not only in Chapter 4 but also in Chapter 5, we state the implication not only for membership queries, but also for equivalence queries, and for all query classes and ontology languages we will encounter.

Lemma 4.12. *The following holds:*

1. *Let $\mathbf{Q} \in \{\text{CQ}, \text{ELIQ}\}$ and $\mathcal{L}$ be an ontology language contained in $\mathcal{ELIHF}_\bot$. If $\mathbf{Q}$ queries are polynomial time learnable under $\mathcal{L}$ ontologies in normal form using membership queries and equivalence queries, the same is true for unrestricted $\mathcal{L}$ ontologies.*

2. *If ELQs are polynomial time learnable under $\mathcal{EL}^r$ ontologies in normal form using membership queries and equivalence queries, then the same is true for unrestricted $\mathcal{EL}^r$ ontologies.*

Proof. Assume that $\mathbf{Q}$ queries are polynomial time learnable under $\mathcal{L}$ ontologies in normal form. Then, there is a learning algorithm $\mathbf{A}$ that takes as input an $\mathcal{L}$ ontology on normal form and a signature Σ. We will use $\mathbf{A}$ to formulate a learning algorithm $\mathbf{A}'$ for $\mathbf{Q}$ queries under unrestricted $\mathcal{L}$ ontologies.

The algorithm $\mathbf{A}'$ takes as input an $\mathcal{L}$ ontology $\mathcal{O}$ and a signature Σ. It first computes the ontology $\mathcal{O}'$ in normal form as per Lemma 4.8 introducing the fresh concept names $\mathbf{X}$ such that $\Sigma \cap \mathbf{X} = \emptyset$. It then starts running $\mathbf{A}$ on input $\mathcal{O}'$ and $\Sigma \cup \mathbf{X}$. The membership queries and equivalence queries that $\mathbf{A}$ makes, cannot be directly passed on to the teacher of $\mathbf{A}'$, as $\mathbf{A}$ expects the teacher to work with the ontology $\mathcal{O}'$, but the teacher answers questions under $\mathcal{O}$. Therefore, $\mathbf{A}'$ modifies the membership queries and equivalence queries done by $\mathbf{A}$ as follows.

Whenever $\mathbf{A}$ asks a membership query with the example $(\mathcal{A}, \overline{a})$, $\mathbf{A}'$ constructs the example $(\mathcal{A}_\Sigma, \overline{a})$ and returns the result of a membership query with the example $(\mathcal{A}_\Sigma, \overline{a})$ to $\mathbf{A}$. Since q_T is guaranteed to only use symbols from Σ, it follows from Point 1 of Lemma 4.11 that $\mathcal{A}, \mathcal{O}' \models q_T(\overline{a})$ if and only if $\mathcal{A}_\Sigma, \mathcal{O} \models q_T(\overline{a})$. Hence, $\mathbf{A}$ receives the correct response to its membership query.

Whenever $\mathbf{A}$ asks an equivalence query with hypothesis q_H, the algorithm $\mathbf{A}'$ constructs the query q_{H_Σ} and instead asks an equivalence query with q_{H_Σ} to its teacher. Lemma 4.10 ensures that q_{H_Σ} belongs to the query class that can be used in

equivalence queries. If the response is a counterexample $(\mathcal{A}, \overline{a})$, **A** instead forwards $(\mathcal{A}|_\Sigma, \overline{a})$ to **A'**.

Since q_T uses only symbols from Σ, $q_{T_\Sigma} = q_T$ and therefore $\mathcal{A}, \mathcal{O} \models q_T(\overline{a})$ if and only if $\mathcal{A}|_\Sigma, \mathcal{O}' \models q_T(\overline{a})$ by Lemma 4.11 Point 2. Likewise, $\mathcal{A}, \mathcal{O} \models q_{H_\Sigma}(\overline{a})$ if and only if $\mathcal{A}|_\Sigma, \mathcal{O}' \models q_H(\overline{a})$ by Lemma 4.11 Point 2. Hence, $(\mathcal{A}|_\Sigma, \overline{a})$ is a correct counterexample to the equivalence query asked by **A**.

If the response to the equivalence query is *yes*, then **A'** forwards this to **A**. This is a correct response to the equivalence query asked by **A** for the same reasons. □

Inspection of the proof of Lemma 4.12 reveals that this reduction preserves the number of membership and equivalence queries done by the learning algorithm. Hence, a version of Lemma 4.12 with polynomial time learnability replaced with polynomial query learnability also holds.

Lemma 4.12 is an example of a reduction from one learning problem to another. For the PAC learning model, these reductions are called *prediction preserving reductions* and were formalized by Pitt and Warmuth [PW90]. Since we will not encounter them often, we avoid defining these reductions in general for our learning settings.

4.3 Frontiers of Queries

In order to approach a target ELIQ q_T from an ELIQ q with $q \subseteq_\mathcal{O} q_T$, we need a way to efficiently search all *generalizations* of q for a query q' such that $q \subseteq_\mathcal{O} q' \subseteq_\mathcal{O} q_T$. In general, a query can have an infinite number of generalizations, but to find a suitable q', we only need to consider the *most-specific* generalizations of q'. This notion is formalized by *frontiers* of queries.

Definition 4.13 (Frontier)**.** Let $\mathcal{O}$ be an ontology and q an ELIQ. A *frontier of q under $\mathcal{O}$* is a set F of ELIQs, such that

1. for all $p \in F$, $q \subseteq_\mathcal{O} p$ and $p \not\subseteq_\mathcal{O} q$;
2. for all ELIQs q' with $q \subseteq_\mathcal{O} q'$ and $q' \not\subseteq_\mathcal{O} q$ there is a $p \in F$ such that $p \subseteq_\mathcal{O} q'$.

Frontiers are very close to downward refinement operators. Definition 4.13 implies that if we have a query q with $q \subseteq_\mathcal{O} q_T$ and $q_T \not\subseteq_\mathcal{O} q$ and F is a frontier of q under $\mathcal{O}$, then there must be a query $p \in F$ such that $p \subseteq_\mathcal{O} q_T$.

Example 4.14. Consider the ELIQs q and p_1, p_2, p_3 in Figure 4.5 and $\mathcal{O} = \emptyset$. It holds that $q \subseteq_\emptyset p_i$ and $p_i \not\subseteq_\emptyset q$ for $1 \leq i \leq 3$. The set $\{p_1, p_2\}$ is a frontier of q under $\mathcal{O}$. The set $\{p_1, p_2, p_3\}$ is also a frontier of q under $\mathcal{O}$, but this frontier is not $\subseteq$-minimal, as p_3 can be removed. The set $\{p_1, p_3\}$ is not a frontier of q, as there is no ELIQ p' in

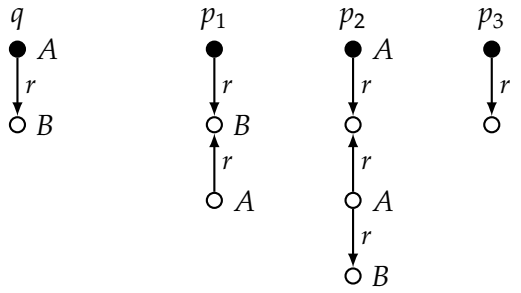

Figure 4.5: The queries $\{p_1, p_2\}$ and $\{p_1, p_2, p_3\}$ are frontiers of the ELIQ q under the empty ontology.

$\{p_1, p_3\}$ such that $p' \subseteq_\varnothing p_2$. It is interesting to observe that the two ELIQs in $\{p_1, p_2\}$ intuitively correspond to the two ways we can generalize q at the root. First, p_1 captures all queries p' with $q \subseteq_\varnothing p'$ that do not have the A atom. Second, p_2 captures all queries p' with $q \subseteq_\varnothing p'$ that generalize the r-successor in q, and thereby remove the B-atom.

Definition 4.13 only applies to ELIQs, but it is of course also possible to define the notion of frontier for other query classes. In fact, there are many choices as the query class of q, of the queries in F, and of the queries considered in Point 2 need not be the same. In some contexts, it makes sense to consider stronger notions of frontiers, where Point 2 should hold for all CQs. Occasionally, we will mention frontiers of ELIQs that consist of CQs, or frontiers of CQs.

We can use frontiers to construct unique characterizations of queries.

Proposition 4.15. *Let $\mathcal{O}$ be an ontology, q an ELIQ and F a frontier of q under $\mathcal{O}$. Then, the positively labeled example $(\mathcal{A}_q, a_q)$ together with the set of all negatively labeled examples $(\mathcal{A}_p, a_p)$ for $p \in F$ form a unique characterization of q under $\mathcal{O}$.*

Proof. Let q' be an ELIQ that fits the examples under $\mathcal{O}$. Since $\mathcal{A}_q, \mathcal{O} \models q'(a_q)$, it follows that $q \subseteq_\mathcal{O} q'$. From F being a frontier of q under $\mathcal{O}$, $q \subseteq_\mathcal{O} q'$ and $\mathcal{A}_p, \mathcal{O} \not\models q'(a_p)$ for all $p \in F$, it then follows that $q' \subseteq_\mathcal{O} q$. Therefore, $q' \equiv_\mathcal{O} q$. □

Proposition 4.15 is why it is unsurprising that CQs also do not possess finite frontiers with regard to all CQs, even under the empty ontology [tCD22]. Fortunately, the situation for ELIQs is better. Finite frontiers of ELIQs in the sense of Definition 4.13 are known to always exist.

Proposition 4.16 ([tCD22])**.** *Let q be an ELIQ. Then, a frontier of q under the empty ontology can be computed in polynomial time.*

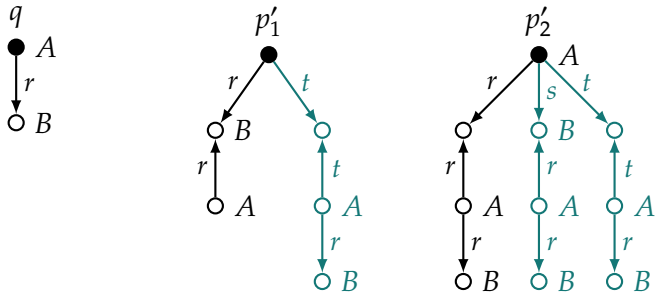

Figure 4.6: The set $\{p_1', p_2'\}$ is a frontier of q under $\mathcal{O} = \{r \sqsubseteq s,\ A \sqsubseteq \exists t.\top\}$. The marked atoms of p_1' and p_2' are not in p_1 or p_2 from Figure 4.5 respectively.

The construction of a frontier of an ELIQ q under the empty ontology consists of two steps. First, a set of ELIQs is inductively computed that represents all possible ways to generalize q. Then, the ELIQs in this set are made most-specific by attaching copies of q. We will use the same strategy to generalize Proposition 4.16 to the case with $\mathit{DL\text{-}Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontologies.

Example 4.17. To see the impact ontologies can have on frontiers, consider the ELIQs displayed in Figure 4.5 under the $\mathit{DL\text{-}Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontology

$$\mathcal{O} = \{r \sqsubseteq s,\ A \sqsubseteq \exists t.\top\}.$$

Under $\mathcal{O}$, the set $\{p_1, p_2, p_3\}$ is no longer a frontier of q: For the ELIQ $p_4(x_0) \leftarrow A(x_0) \wedge s(x_0, x_1) \wedge B(x_1)$, it holds that $q \subseteq_{\mathcal{O}} p_4$ and $p_4 \not\subseteq_{\mathcal{O}} q$, but there is no $p_i \in \{p_1, p_2, p_3\}$ with $p_i \subseteq_{\mathcal{O}} p_4$. Even the set $\{p_1, p_2, p_3, p_4\}$ does not suffice as a frontier of q under $\mathcal{O}$, as ELIQs that use the role name t have also to be taken into consideration.

However, it is possible to construct a frontier under $\mathcal{O}$ by taking concept inclusions and role inclusions in $\mathcal{O}$ into account during the construction of a frontier. Figure 4.6 shows the same query q as Figure 4.5 and extended versions of p_1 and p_2, where red color denotes atoms that were added. Intuitively, the set of all ELIQs p such that $q \subseteq_{\mathcal{O}} p$ is a superset of the set of all ELIQs p' such that $q \subseteq_{\varnothing} p'$, and p_1, p_2 have to contain additional atoms to capture these additional ELIQs. The intuition from Example 4.14 still applies though: p_1' captures all generalizations of q that remove the A-atom at the root, and p_2' captures all generalizations of q that generalize the r-successor. These extensions make $\{p_1', p_2'\}$ a frontier of q under $\mathcal{O}$.

Unfortunately, it does not always suffice to extend the queries in a frontier under the empty ontology, as $\mathit{DL\text{-}Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontologies can also reduce the number of ways an ELIQ can be generalized. Consider $\mathcal{O}' = \{\exists r.\top \sqsubseteq A\}$. The set $\{p_1, p_2\}$ is not a frontier of q under $\mathcal{O}'$, as $p_1 \equiv_{\mathcal{O}'} q$. Indeed, under $\mathcal{O}'$ the set $\{p_2\}$ is already a frontier of q.

Minimality

In order to simplify the construction of a frontier of an ELIQ q, we will assume that q does not use superfluous existential variables. The ELIQ $q(x_0) \leftarrow r(x_0, x_1) \wedge r(x_0, x_2)$ is equivalent to the ELIQ $p(x_0) \leftarrow r(x_0, x_1)$, but uses the additional variable x_2. The additional variable (and the atoms it occurs in) can be removed without affecting the answers to q.

In the case without ontologies, this is formalized by q being a *core*, meaning that every homomorphism from q to itself is surjective. This does not suffice here, as ontologies may introduce additional redundancies through concept inclusions. Instead, we will formalize this as follows.

For an ABox $\mathcal{A}$ and a set $S \subseteq \mathsf{ind}(\mathcal{A})$ we use $\mathcal{A}|_S$ to denote the restriction of $\mathcal{A}$ to only individual names in S. Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology, q a CQ and x an existential variable of q. The CQ $q^{-\mathcal{O}x}$ is then obtained by removing all atoms that mention x from q, and then adding, for all variables $y \in \mathsf{var}(q) \setminus \{x\}$ and concept names A with $\mathcal{A}_q, \mathcal{O} \models A(y)$ but $\mathcal{A}_q|_{\mathsf{var}(q)\setminus\{x\}}, \mathcal{O} \not\models A(y)$, the atom $A(y)$.

Definition 4.18 (Minimal CQ)**.** Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology and p, q CQs. The CQ q is $(p, \mathcal{O})$-*minimal* if there is no existential variable $x \in \mathsf{var}(q)$ such that $q^{-\mathcal{O}x} \subseteq_{\mathcal{O}} p$.

Definition 4.18 allows us to formulate a suitable notion of q being *minimal* under an ontology, namely that q is $(q, \mathcal{O})$-minimal. Note that if a CQ q is $(p, \mathcal{O})$-minimal for some p with $q \subseteq_{\mathcal{O}} p$, then q is also $(q, \mathcal{O})$-minimal. In this section, we only need this notion of minimality for ELIQs and *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ ontologies, but we will also use it in later sections, which is why Definition 4.18 is formulated for all CQs and $\mathcal{ELIHF}_\bot$ ontologies. Note that the negative condition $\mathcal{A}_q|_{\mathsf{var}(q)\setminus\{x\}}, \mathcal{O} \not\models A(y)$ in the construction of $q^{-\mathcal{O}x}$ is not required for the following results, is and only included as a practical concern to avoid unnecessary work by adding concept name atoms that are later removed.

The central technical properties of $(p, \mathcal{O})$-minimal queries, that we heavily rely on in our frontier construction, are the following:

Lemma 4.19. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form and $p(\overline{y}_0)$ and $q(\overline{x}_0)$ CQs such that q is satisfiable under $\mathcal{O}$ and p is rooted. If h is a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{y}_0) = \overline{x}_0$ and there is a variable $x \in \mathsf{var}(q)$ with $x \notin \mathsf{img}(h)$, then h is also a homomorphism from p to $\mathcal{U}_{q^{-\mathcal{O}x},\mathcal{O}}$.*

Proof. Let h be a homomorphism as required with $x \notin \mathsf{img}(h)$. Set $q' = q^{-\mathcal{O}x}$. Note that since p is rooted, there is no variable $y \in \mathsf{var}(p)$ that is mapped to a trace starting with x in $\mathcal{U}_{q,\mathcal{O}}$. Now, let x' be a variable in q with $x' \neq x$ and $x'RM$ be a trace of length one in $\mathcal{U}_{q,\mathcal{O}}$. By construction of $\mathcal{U}_{q,\mathcal{O}}$ and the normal form of $\mathcal{O}$, $x'RM \in \Delta^{\mathcal{U}_{q,\mathcal{O}}}$ implies that there is a set of concept names M' such that $\mathcal{A}_q, \mathcal{O} \models \bigsqcap M(x')$ and

$\mathcal{O} \models \bigsqcap M' \sqsubseteq \exists R. \bigsqcap M$. By construction of $q' = q^{-o x}$ it follows that $\mathcal{A}_{q'}, \mathcal{O} \models \bigsqcap M'(x')$ and therefore, $x'RM$ is also a trace in $\mathcal{U}_{q',\mathcal{O}}$. Hence, all traces in $\mathcal{U}_{q,\mathcal{O}}$ that do not start with x also occur in $\mathcal{U}_{q',\mathcal{O}}$ and h is a well-defined function from $\mathsf{var}(p)$ to $\Delta^{\mathcal{U}_{q',\mathcal{O}}}$.

We verify that h is a homomorphism. Let $A(y)$ be a concept atom in p. Since $h(y) \in A^{\mathcal{U}_{q,\mathcal{O}}}$ and $h(y)$ cannot be a trace starting with x, the construction of q' implies that $h(y) \in A^{\mathcal{U}_{q',\mathcal{O}}}$. This is immediate if $h(y) \in \mathsf{var}(q')$ and follows inductively for all traces. Let $r(y, y')$ be a role atom in p. Since neither $h(y)$ nor $h(y')$ can be a trace starting with x, $(h(y), h(y')) \in r^{\mathcal{U}_{q,\mathcal{O}}}$ implies that $(h(y), h(y')) \in r^{\mathcal{U}_{q',\mathcal{O}}}$. □

Lemma 4.20. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form, $p(\overline{y}_0)$ and $q(\overline{x}_0)$ CQs such that q is $(p,\mathcal{O})$-minimal, q satisfiable under $\mathcal{O}$ and p is rooted.*

For all homomorphisms h from p to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{y}_0) = \overline{x}_0$, $\mathsf{var}(q) \subseteq \mathsf{img}(h)$.

Proof. Let h be a homomorphism as required and assume for showing a contradiction that there is an existential variable $x \in \mathsf{var}(q)$ such that there is no $y \in \mathsf{var}(p)$ with $h(y) = x$. Lemma 4.19 implies that h is also a homomorphism from p to $\mathcal{U}_{q^{-o x},\mathcal{O}}$ with $h(\overline{y}_0) = \overline{x}_0$. Therefore, $q^{-o x} \subsetneq_{\mathcal{O}} p$, contradicting $(p,\mathcal{O})$-minimality of q. □

From Lemma 4.20, we can derive the following property of $(q,\mathcal{O})$-minimal queries, which is a suitable analogue of q being a core.

Lemma 4.21. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form and $q(\overline{x}_0)$ a rooted CQ that is $(q,\mathcal{O})$-minimal and satisfiable under $\mathcal{O}$. For all homomorphisms h from q to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{x}_0) = \overline{x}_0$, $\mathsf{var}(q) = \mathsf{img}(h)$.*

Definition 4.18 directly suggests a procedure to compute a $(q,\mathcal{O})$-minimal query from an ELIQ q: repeatedly check if there is a $x \in \mathsf{var}(q)$ such that $q^{-o x} \subsetneq_{\mathcal{O}} q$. The complexity of this procedure depends on the ontology language. If $\mathcal{A}, \mathcal{O} \models q(a)$ can be decided in polynomial time given q, $\mathcal{O}$, $\mathcal{A}$ and a, then this direct procedure to compute a $(q,\mathcal{O})$-minimal query runs in polynomial time. The ELIQ answering techniques in [Bie+13] for $\mathit{DL\text{-}Lite}_{\mathrm{core}}$ ontologies can be extended to $\mathit{DL\text{-}Lite}^{\mathcal{F}^-}_{\mathrm{core}}$ ontologies, which then implies the following proposition.

Proposition 4.22. *Let $\mathcal{O}$ be a $\mathit{DL\text{-}Lite}^{\mathcal{F}^-}_{\mathrm{core}}$ ontology and q an ELIQ that is satisfiable under $\mathcal{O}$. Then, a $(q',\mathcal{O})$-minimal query q' with $q \equiv_{\mathcal{O}} q'$ can be computed in polynomial time.*

Unfortunately, query answering with ELIQs is known to be NP-hard under $\mathit{DL\text{-}Lite}^{\mathcal{H}}_{\mathrm{core}}$ ontologies [KKZ11]. We conjecture that this results transfers to obtaining $(q,\mathcal{O})$-minimality, meaning that it is not possible to construct equivalent $(q,\mathcal{O})$-minimal ELIQs in polynomial time under $\mathit{DL\text{-}Lite}^{\mathcal{H}}_{\mathrm{core}}$ ontologies, unless P = NP. We will still be able to use the frontier construction as part of a polynomial time learning algorithm under $\mathit{DL\text{-}Lite}^{\mathcal{HF}^-}_{\mathrm{core}}$ ontologies by obtaining $(q,\mathcal{O})$-minimal queries in polynomial time using membership queries.

The Construction of Frontiers

In the remainder of this section, we show the following theorem.

Theorem 4.23. *Let $\mathcal{O}$ be an DL-Lite$_{\text{core}}^{\mathcal{HF}-}$ ontology in normal form and q an ELIQ that is $(q, \mathcal{O})$-minimal and satisfiable under $\mathcal{O}$. Then, a frontier of q under $\mathcal{O}$ can be computed in time polynomial in $\|q\|$ and $\|\mathcal{O}\|$.*

We first describe an explicit construction of frontiers of ELIQs under *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies, then show its correctness, and finally establish that the construction can be done in polynomial time.

Let $\mathcal{O}$ be an *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontology in normal form and $q(x_0)$ an ELIQ that is $(q, \mathcal{O})$-minimal and satisfiable under $\mathcal{O}$. To construct a frontier of q under $\mathcal{O}$, we consider all possible ways to construct most specific generalizations of q. That is, queries q' such that $q \subseteq_{\mathcal{O}} q'$ as well as $q' \not\subseteq_{\mathcal{O}} q$ and for every ELIQ $\widehat{q}$ with $q \subseteq_{\mathcal{O}} \widehat{q}$ and $\widehat{q} \subseteq_{\mathcal{O}} q'$ either $\widehat{q} \equiv_{\mathcal{O}} q$ or $\widehat{q} \equiv_{\mathcal{O}} q'$. We do this in two steps: the actual generalization step, that produces generalizations of q and a compensation step, that carefully adds atoms to make sure that the generalizations are most specific under $\mathcal{O}$.

The construction that follows involves the introduction of fresh variables x, some of which are copies of variables from $\mathsf{var}(q)$. We then use $x^{\downarrow}$ to denote that original variable. We will view $\cdot^{\downarrow}$ as a partial function from the copies to their originals. Recall that q_x denotes the subquery of q that is the subtree below a variable $x \in \mathsf{var}(q)$.

Step 1 Generalize. For each variable $x \in \mathsf{var}(q)$, define a set $F_0(x)$ that contains all ELIQs which can be obtained by starting with $q_x(x)$ and then doing one of the following:

Drop a concept atom

1. choose an atom $A(x) \in q$ such that:
   a) there is no $R(x', x) \in q$ such that $\mathcal{O} \models \exists R^-.\top \sqsubseteq A$,
   b) there is no $B(x) \in q$ such that $\mathcal{O} \models B \sqsubseteq A$ and $\mathcal{O} \not\models A \sqsubseteq B$.
2. remove all concept atoms $B(x)$ with $\mathcal{O} \models A \equiv B$, including $A(x)$.
3. for every concept name B with $\mathcal{O} \models A \sqsubseteq B$ and $\mathcal{O} \not\models B \sqsubseteq A$, add $B(x)$.

Generalize a subquery

1. choose an atom $R(x, y) \in q$ directed away from x_0 and remove $R(x, y)$ as well as all atoms of q_y.
2. for every concept name B with $\mathcal{O} \models \exists R.\top \sqsubseteq B$, add the atom $B(x)$.
3. if $\mathsf{func}(R) \notin \mathcal{O}$, then for each $q'(y) \in F_0(y)$, add a disjoint copy $\widehat{q}'$ of q' and the role atom $R(x, y')$ with y' the copy of y in $\widehat{q}'$.

4. if $\mathsf{func}(R) \in \mathcal{O}$ and $F_0(y) \neq \emptyset$, then choose and add a $q' \in F_0(y)$ and the role atom $R(x, y)$.
5. for every role S with $\mathcal{O} \models R \sqsubseteq S$ and $\mathcal{O} \not\models S \sqsubseteq R$, add a disjoint copy $\widehat{q}_y$ of q_y and the atoms $S(x, y')$, $A(y')$ for all A with $\mathcal{O} \models \exists R^-.\top \sqsubseteq A$ with y' the copy of y in $\widehat{q}_y$.

Note that this is an *inductive* construction, as Point 3 and 4 of *Generalize a subquery* use the set $F_0(y)$. This is well-defined, as the codepth of y is always lower than the codepth of x. The definition of the partial function $\cdot^{\downarrow}$ should be clear in all cases. In Point 3 of *Generalize a subquery*, for example, for every variable z in q' that was renamed to z' in $\widehat{q}'$ set $z'^{\downarrow} = z^{\downarrow}$.

Step 2 Compensate. We construct a frontier F_q of $q(x_0)$ by including, for each $p \in F_0(x_0)$, the ELIQ obtained from p by applying the following two steps.

Step 2 A Consider all variables $x \in \mathsf{var}(p)$, roles R, S and sets of concept names $M = \{A_1, \dots, A_k\}$ such that $x^{\downarrow} \rightsquigarrow^{R}_{q,\mathcal{O}} M$, $\mathcal{O} \models R \sqsubseteq S$, and[3] for all $B(x) \in q$ such that $\mathcal{O} \models \exists S.\top \sqsubseteq B$, $B(x) \in p$. Add the atoms $S(x, z), A_1(z), \dots, A_k(z), R(x', z)$ where z and x' are fresh variables. Set $x'^{\downarrow} = x^{\downarrow}$ and leave $z^{\downarrow}$ undefined. Add a disjoint copy $\widehat{q}$ of q and glue the copy of $x^{\downarrow}$ to x'.

Step 2 B This step is iterative. For bookkeeping, we mark atoms $R(x, y) \in p$ to be processed in the next round of iteration.

To start, consider every $R(x, y) \in p$ directed *towards* x_0 such that $x^{\downarrow}$ and $y^{\downarrow}$ are defined and such that $R(x, y)$ was not added in Step 2 A. For every role S such that $\mathcal{A}_q, \mathcal{O} \models S(x^{\downarrow}, y^{\downarrow})$ and if $S = R$, $\mathsf{func}(R) \notin \mathcal{O}$, add the atom $S(x, y')$ where y' is a fresh variable with $y'^{\downarrow} = y^{\downarrow}$ and mark it. Note that all added atoms $S(x, y')$ are directed *away from* x_0.

Then repeatedly choose a marked atom $R(x, y) \in p$ and remove its mark. If $\mathsf{func}(R^-) \notin \mathcal{O}$ or q contains no atom of the form $R(y^{\downarrow}, z)$, then add a disjoint copy $\widehat{q}$ of q and glue a copy of $y^{\downarrow}$ in $\widehat{q}$ to y. Otherwise, do the following:

1. add $A(y)$ whenever $\mathcal{A}_q, \mathcal{O} \models A(y^{\downarrow})$.
2. for all atoms $S(y^{\downarrow}, z) \in q$ and roles S' with $\mathcal{O} \models S \sqsubseteq S'$ such that $S' \neq R^-$, add an atom $S'(y, z')$ where z' is a fresh variable. Set $z'^{\downarrow} = z$ and mark the atom $S'(y, z')$.
3. for all roles S and sets $M = \{A_1, \dots, A_k\}$ such that $y^{\downarrow} \rightsquigarrow^{S}_{q,\mathcal{O}} M$, add the atoms $S(y, u), A_1(u), \dots, A_k(u), S^-(u, y')$ where u and y' are fresh variables. Set $y'^{\downarrow} = y^{\downarrow}$ and mark the atom $S^-(u, y')$.

[3] This last condition avoids re-adding a concept name that may have been dropped in Step 1.

This finishes the construction of F_q.

Example 4.24. Consider the ELIQ $q(x_0) \leftarrow A(x_0) \wedge r(x_0, x_1) \wedge B(x_1)$ and the $\mathit{DL\text{-}Lite}^{\mathcal{HF}-}_{\text{core}}$ ontology

$$\mathcal{O} = \{r \sqsubseteq s,\ t \sqsubseteq u,\ A \sqsubseteq \exists t.\top,\ \exists t.\top \sqsubseteq A,\ \mathsf{func}(r)\}.$$

Figure 4.7 shows the steps of the construction of a frontier of q under $\mathcal{O}$. The construction introduces many new variables. To understand the origin of the new variables, Figure 4.7 shows a name that refers to the original variable they are a copy of. The set $F_0(x_1)$ contains only the empty query. Hence, the result of Step 1 $F_0(x_0)$ is $\{p_1, p_2\}$. Note that because $r \sqsubseteq s \in \mathcal{O}$ the variable x_0 has an extra s-successor in p_1.

The application of Step 2 A to $F_0(x_0)$ results in the set $\{p_1', p_2'\}$, the added atoms are marked in blue. We slightly simplify p_1' and p_2' by leaving out superfluous u-successors that are attached by Step 2 A, but already homomorphically contained in the marked atoms. Note that in the construction of p_1', no t-successor is attached to x_0 as $\mathcal{O} \models \exists t.\top \sqsubseteq A$. Instead, a u-successor is added, since $t \sqsubseteq u \in \mathcal{O}$ and $\mathcal{O} \not\models \exists u.\top \sqsubseteq A$.

The set $F_q = \{p_1'', p_2''\}$ is then the result of Step 2 B. Again, additions are marked with colors. The start of Step 2 B produces new s-predecessors and new r-predecessors (blue), that are then marked. As $\mathsf{func}(s) \notin \mathcal{O}$, copies of q are then attached to the s-predecessors when they are processed (blue). The r-predecessors cannot be handled in this way, as $\mathsf{func}(r) \in \mathcal{O}$. Instead, an s-successor is added and marked in Point 2 of Step 2 B (yellow) and a t-successor as well as a copy of q is added in Point 3 of Step 2 B (purple). When the marked s-successor is processed, a copy of q is attached (yellow). This completes this example, $\{p_1'', p_2''\}$ is a frontier of q under $\mathcal{O}$.

Note that p_1'' is not $(p_1'', \mathcal{O})$-minimal, as most of the atoms introduced in Step 2 B can be removed. The query p_2'' is also not $(p_2'', \mathcal{O})$-minimal, for similar reasons. The construction could be modified to avoid these superfluous additions in Step 2, but we would rather avoid the additional complexity, as Step 2 is complicated enough.

We continue with showing that F_q is indeed a frontier of q under $\mathcal{O}$. We begin with showing that every query in F_q is satisfiable under $\mathcal{O}$.

Lemma 4.25. *Every $p \in F_q$ is satisfiable under $\mathcal{O}$.*

Proof. Let p be a query from F_q. It can easily be shown that if q is satisfiable under $\mathcal{O}$, then there is a model of $\mathcal{A}_p$ that satisfies all concept inclusions and role disjointness constraints in $\mathcal{O}$, for example, by noting that $\cdot^{\downarrow}$ can be extended to a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$. It remains to show that $\mathcal{A}_p$ also satisfies all functionality constraints in $\mathcal{O}$. For this, we will consider all steps of the construction of p that add role atoms.

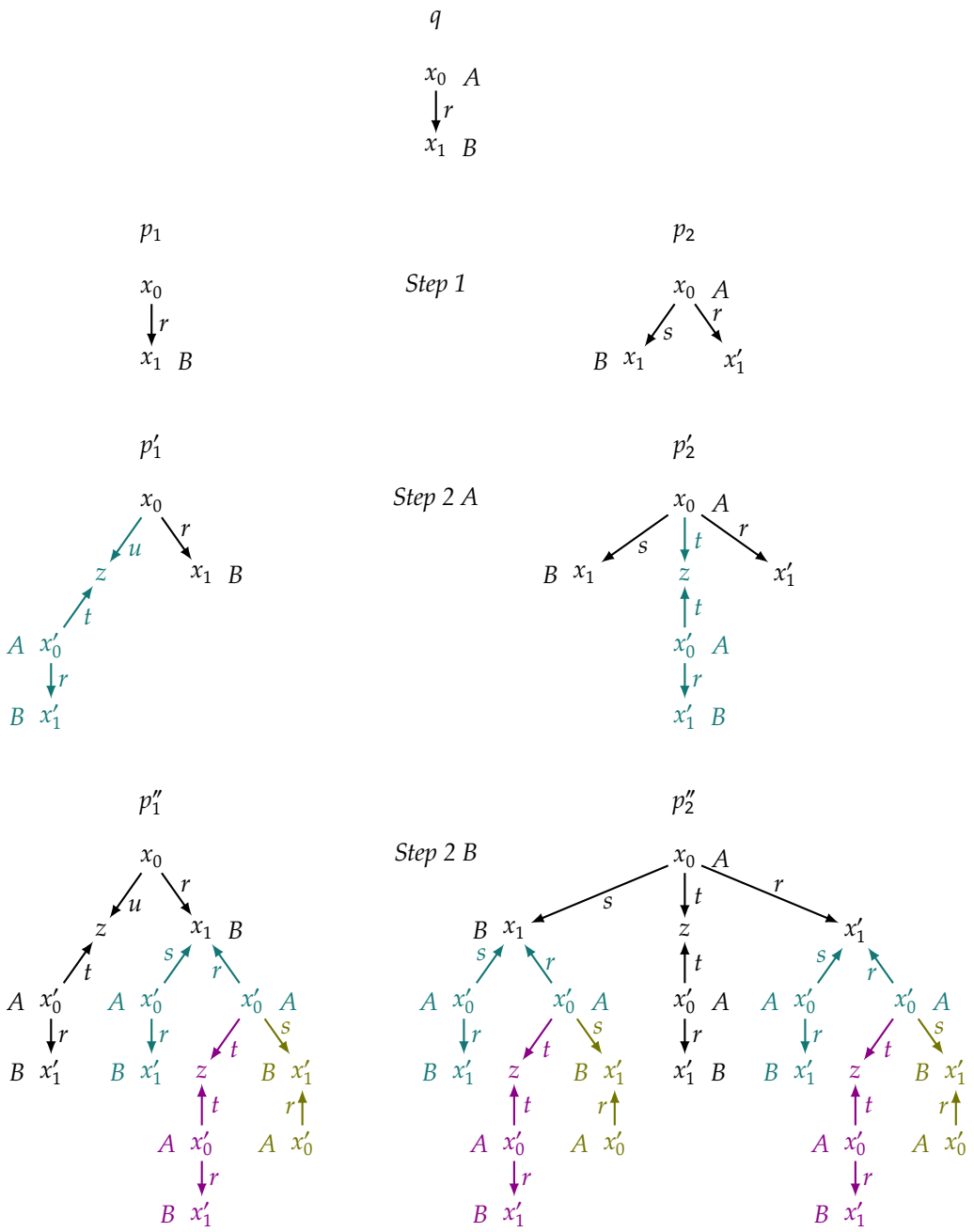

Figure 4.7: The steps of the construction of a frontier of the ELIQ $q(x_0) \leftarrow A(x_0) \wedge r(x_0, x_1) \wedge B(x_1)$ under the ontology $\mathcal{O} = \{r \sqsubseteq s,\ t \sqsubseteq u,\ A \equiv \exists t.\top,\ \mathsf{func}(r)\}$.

In Step 1, Points 3 and 4 of generalizing a subquery add role atoms and attach subqueries. Point 3 only applies if $\mathsf{func}(R) \notin \mathcal{O}$, Point 4 adds a single R atom, so every functionality constraint is satisfied, as Point 1 removed an R atom. Point 5 adds only role atoms $S(x, y')$ if $\mathcal{O} \models R \sqsubseteq S$ and $\mathcal{O} \not\models S \sqsubseteq R$, it follows that $\mathsf{func}(S) \notin \mathcal{O}$.

In Step 2 A, note that since $x^{\downarrow} \rightsquigarrow^{R}_{q,\mathcal{O}} M$, either $\mathsf{func}(R) \notin \mathcal{O}$ or there is no variable $y \in \mathsf{var}(q)$ such that $\mathcal{A}_q, \mathcal{O} \not\models R(x^{\downarrow}, y)$. Furthermore, it follows that $\mathsf{func}(S^-) \notin \mathcal{O}$. Thus, if $\mathcal{O} \models R \equiv S$, then the new atoms $S(x, z), R(x', z)$, as well as the attached copy of q satisfy all functionality constraints. If $\mathcal{O} \not\models S \sqsubseteq R$, it then follows that $\mathsf{func}(S) \notin \mathcal{O}$, and therefore all functionality constraints are satisfied.

At the start of Step 2 B, the atom $S(x, y')$ is only added if it satisfies all functionality constraints. The same applies to the iterative step of Step 2 B. The copy of q is only attached if it does not violate a functionality constraint. Point 2 only attaches atoms from q that do not violate the functionality constraint $\mathsf{func}(R^-)$, and the condition $y^{\downarrow} \rightsquigarrow^{S}_{q,\mathcal{O}} M$ of Point 3 ensures that no functionality constraint is violated. □

We now move on to show that F_q is indeed a frontier of q under $\mathcal{O}$.

Lemma 4.26. *F_q is a frontier of $q(x_0)$ under $\mathcal{O}$.*

Proof. We show that F_q fulfills the two conditions of frontiers. For the first condition, let $p(x_0)$ be a query from F_q. We begin by showing $q \subseteq_{\mathcal{O}} p$. From Lemma 4.25 and satisfiability of q, it follows that p is also satisfiable under $\mathcal{O}$. Hence, by Lemma 3.7 it suffices to show $p(x_0) \to \mathcal{U}_{q,\mathcal{O}}, x_0$.

The mapping $\cdot^{\downarrow}$ is already almost the required homomorphism. We extend the mapping $\cdot^{\downarrow}$ to be defined on all variables of p by considering the yet unmapped variables added in Step 2 A and Step 2 B of the construction. Let z be such a fresh variable added in Step 2 A for $x \in \mathsf{var}(p)$, roles R, S and sets concept names M. Then $x^{\downarrow} \rightsquigarrow^{R}_{q,\mathcal{O}} M$ and by construction of $\mathcal{U}_{q,\mathcal{O}}$, there is a trace $x^{\downarrow}RM \in \Delta^{\mathcal{U}_{q,\mathcal{O}}}$. Set $z^{\downarrow} = x^{\downarrow}RM$. Similarly for variables added in Step 2 B. Let u be a fresh variable added in Point 3 of Step 2 B for the variable y, role S and set of concept names M. Set $u^{\downarrow} = y^{\downarrow}SM$. Now $\cdot^{\downarrow}$ is defined on all variables of p and, by construction of p, it is a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$ with $x_0^{\downarrow} = x_0$ as required.

For $p \not\subseteq_{\mathcal{O}} q$, we first show the following claim.

Claim 1. For all $x \in \mathsf{var}(q)$ and $p(x) \in F_0(x)$, $p \not\subseteq_{\mathcal{O}} q_x$.

Proof of Claim 1. We show the claim by induction on the codepth of x in q, matching the inductive construction of F_0.

In the induction start, x has codepth 0. Then, by definition of codepth, there is no role atom $R(x, y) \in q$ that is directed away from x_0 and all $p \in F_0(x)$ must be obtained by dropping a concept atom. Let $p(x)$ be a query in $F_0(x)$ that is obtained

by dropping the concept atom $A(x) \in q_x$. Then, by choice of $A(x)$, there is no $B(x) \in p$ with $\mathcal{O} \models B \sqsubseteq A$ and no $R(x, x') \in p$ with $\mathcal{O} \models \exists R.\top \sqsubseteq A$. Hence, $A(x) \in q_x$ and $A(x) \notin \mathcal{U}_{p,\mathcal{O}}$, therefore $q_x(x) \nrightarrow \mathcal{U}_{p,\mathcal{O}}, x$ and thus $p \not\subseteq_{\mathcal{O}} q_x$.

In the induction step, let x have codepth > 0, let $p(x)$ be a query in $F_0(x)$ and assume that the claim holds for all variables of q with smaller codepth. Let $\cdot^{\downarrow}$ be the extension of the original $\cdot^{\downarrow}$ for p to a homomorphism from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q_x,\mathcal{O}}$ that exists by Lemma 3.8. If p is obtained by dropping a concept atom, then the same argument as in the induction start yields $p \not\subseteq_{\mathcal{O}} q_x$. If p is obtained by generalizing a subquery attached to a role atom $R(x, y) \in q_x$, assume for contradiction that there is a homomorphism h from q_x to $\mathcal{U}_{p,\mathcal{O}}$ with $h(x) = x$. From h we construct a homomorphism h' from q to $\mathcal{U}_{q,\mathcal{O}}$ with $h'(x_0) = x_0$ by setting $h'(z) = h(z)^{\downarrow}$ for all $z \in \mathsf{var}(q_y)$ and $h'(z) = z$ for all $z \notin \mathsf{var}(q_y)$. We will use h' to show that q cannot be $(q, \mathcal{O})$-minimal using Lemma 4.21, leading to a contradiction

The homomorphism h must map y to an R-successor of x in $\mathcal{U}_{p,\mathcal{O}}$, we distinguish the following cases.

- $h(y)$ is a $z \in \mathsf{var}(p)$ with $z^{\downarrow} \neq y$.

  Since $z^{\downarrow} = z$ by definition of $\cdot^{\downarrow}$, and since $h'(z) = z$ by definition of h', $h'(y) = h'(z) = z$ and h' is a non-injective homomorphism. This contradicts $(q, \mathcal{O})$-minimality of q using Lemma 4.21.

- $h(y)$ is a trace $h(x)SM \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$ for some role S with $\mathcal{O} \models S \sqsubseteq R$ and set of concept names M.

  Then, if $h'(y)$ is also a trace, there must be a $y' \in \mathsf{var}(q)$ with $y' \notin \mathsf{img}(h')$, and h' cannot be a surjective homomorphism. This contradicts $(q, \mathcal{O})$-minimality of q using Lemma 4.21.

  If $h'(y)$ is not a trace, but a successor y' of x with $y' \neq y$, then by definition of h', $h'(y') = h'(y) = y'$ and h' is not an injective homomorphism. Again, this contradicts $(q, \mathcal{O})$-minimality of q using Lemma 4.21.

  If $h'(y) = y$ and there is a $y' \in \mathsf{var}(q_y)$ with $h'(y') = x$, then $h'(y') = h'(x) = x$, and again h' is not an injective homomorphism. Again, this contradicts $(q, \mathcal{O})$-minimality of q using Lemma 4.21.

  If $h'(y) = y$ and there is no $y' \in \mathsf{var}(q_y)$ with $h'(y') = x$, then we show a contradiction to $(q, \mathcal{O})$-minimality of q by constructing a homomorphism h'' from q to $\mathcal{U}_{q^{-}\mathcal{O}^{y},\mathcal{O}}$ with $h''(x_0) = x_0$ Note that by construction of h', $h'(z) = y$ implies $z = y$.

  Since $(h(x)SM)^{\downarrow} = y$, there is no trace $xSM \in \Delta^{\mathcal{U}_{q,\mathcal{O}}}$ and $\mathcal{O} \models R \equiv S$. But, since $h(x)SM \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$ it must be that $h(x) \rightsquigarrow^{S}_{p,\mathcal{O}} M$ and thus $\mathcal{A}_p, \mathcal{O} \models \exists R. \bigsqcap M(h(x))$

and $\mathcal{A}_q, \mathcal{O} \models \exists R. \bigsqcap M(x)$. However, $x \not\rightsquigarrow^S_{q,\mathcal{O}} M$ because $\mathcal{O} \models R \equiv S$, $R(x,y) \in q$ and $\mathcal{A}_q, \mathcal{O} \models \bigsqcap M(y)$. Since $R(x,y) \notin q^{-\mathcal{O}^y}$ it follows by construction of $q^{-\mathcal{O}^y}$ and the normal form of $\mathcal{O}$ that $x \rightsquigarrow^S_{q^{-\mathcal{O}^y}} M$. Therefore, there is a trace $xSM \in \Delta^{\mathcal{U}_{q^{-\mathcal{O}^y},\mathcal{O}}}$.

We construct the new homomorphism h'' by setting $h''(z) = h'(z)$ for all $z \in \mathsf{var}(q) \setminus \mathsf{var}(q_y)$ and $h''(z) = xSMR_2M_2 \dots R_nM_n$ for all $z \in \mathsf{var}(q_y)$ with $h(z) = h(x)SMR_2M_2 \dots R_nM_n$.

- $h(y)$ is the root y' of a query $p' \in F_0(y)$ that was added in Point 3 or Point 4 of generalizing a subquery.

  Then, by the induction hypothesis, $q_y(y) \twoheadrightarrow \mathcal{U}_{p',\mathcal{O}}, y'$ for all $p' \in F_0(y)$. By the normal form of $\mathcal{O}$ and definition of $\mathcal{U}_{p,\mathcal{O}}$, the subtree below y' in $\mathcal{U}_{p,\mathcal{O}}$ contains $\mathcal{U}_{p',\mathcal{O}}$, but may not be identical to $\mathcal{U}_{p,\mathcal{O}}$ since $(x,y') \in R^{\mathcal{U}_{p,\mathcal{O}}}$ but $(x,y') \notin R^{\mathcal{U}_{p',\mathcal{O}}}$. In particular, there may be concept names A with $y' \in A^{\mathcal{U}_{p,\mathcal{O}}}$, $y' \notin A^{\mathcal{U}_{p',\mathcal{O}}}$ and $\mathcal{O} \models \exists R^-.\top \sqsubseteq A$, or traces of the form $y'SM\cdots \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$ with $y'SM\cdots \notin \Delta^{\mathcal{U}_{p',\mathcal{O}}}$ and $\mathcal{O} \models \exists R^-.\top \sqsubseteq \exists S.\top$.

  Consequently, one of the following must be true:

  - there is a $y'' \in \mathsf{var}(q_y)$ with $h(y'') = x$. Then, by definition of h', $h'(y'') = h'(x) = x$ and h' is not an injective homomorphism. This contradicts $(q,\mathcal{O})$-minimality of q using Lemma 4.21.
  - there is a $y'' \in \mathsf{var}(q_y)$ with $h(y'') = y'$ and $A(y'') \in q_y$ for some concept name A with $y' \notin A^{\mathcal{U}_{p',\mathcal{O}}}$. Then $y'' \neq y$, by choice of the concept atom during the dropping of a concept atom. Hence, $h'(y'') = h'(y) = y$ implies that h' is not an injective homomorphism. Again, this contradicts $(q,\mathcal{O})$-minimality of q using Lemma 4.21.
  - there is a $y'' \in \mathsf{var}(q_y)$ with $h(y'') = y'SM$ for a trace $y'SM \notin \Delta^{\mathcal{U}_{p',\mathcal{O}}}$.

    First, observe that y'' must be a successor of y, and the entire tree below y'' must be mapped by h into traces below $y'SM$. Otherwise, there is a $y''' \in q_y$ with $y''' \neq y$ and $h(y''') = h(y) = y'$, which implies that h' is not injective, contradicting $(q,\mathcal{O})$-minimality of q.

    Then, if $y'SM^{\downarrow}$ is a trace, then so is $h'(y'')$, contradicting $(q,\mathcal{O})$-minimality of q by Lemma 4.21. If $y'SM^{\downarrow}$ is not a trace, but a successor z of y in q, then there is no variable in q_y that is mapped by h to a variable $z' \in \mathsf{var}(p)$ with $z'^{\downarrow} = z$, since otherwise h' is not injective. Then, we can construct a homomorphism h'' from q to $\mathcal{U}_{q^{-\mathcal{O}^z},\mathcal{O}}$ with $h''(x_0) = x_0$, contradicting $(q,\mathcal{O})$-minimality of q. Set $h''(x) = h'(x)$ for all $x \notin \mathsf{var}(q_{y''})$ and set $h''(x) = ySMt$ for all $x \in \mathsf{var}(q_{y''})$ with $h(x) = y'SMt$.

Note that $h(y)$ cannot be a variable introduced in Point 5 of generalizing a subquery, as at that point only S-successors of x that satisfy $\mathcal{O} \not\models S \sqsubseteq R$ are introduced. This completes the proof of Claim 1.

Now, assume for contradiction that $p \subsetneq_{\mathcal{O}} q$. Then, there is a homomorphism h from q to $\mathcal{U}_{p,\mathcal{O}}$ with $h(x_0) = x_0$. Let $\cdot^{\downarrow}$ be the extension of the original $\cdot^{\downarrow}$ for p to a homomorphism from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$ which exists by Lemma 3.8. We compose h and $\cdot^{\downarrow}$ to construct a homomorphism h' from q to $\mathcal{U}_{q,\mathcal{O}}$ with $h'(x_0) = x_0$. By Claim 1, there is no homomorphism that maps q entirely into $\mathcal{U}_{p',\mathcal{O}}$ for any $p' \in F_0(x_0)$. Hence, there must be an $x \in \mathsf{var}(q)$ such that $h(x)$ is a fresh variable added in Step 2. By definition of that step and since q is connected, we may distinguish the following two cases:

- $h(x)$ is a fresh variable z added in Step 2 A. Then, since $x^{\downarrow} \rightsquigarrow^{R}_{q,\mathcal{O}} M$, $z^{\downarrow}$ and this $h'(x)$ is a trace in $\mathcal{U}_{q,\mathcal{O}}$. Hence, h' contradicts $(q,\mathcal{O})$-minimality of q by Lemma 4.21.
- $h(x)$ is a fresh variable y' added at the start of Step 2 B for the role atom $R(x,y) \in p$ with $y^{\downarrow} = y'^{\downarrow}$. Then, since q is connected, there must be a predecessor x' of x with $h(x') = y$. Hence, $h'(x) = h'(x') = y^{\downarrow}$, contradicting $(q,\mathcal{O})$-minimality of q by Lemma 4.21.

This completes the proof that the first condition of frontiers holds.

For the second condition of frontiers, let $q'(x_0)$ be an ELIQ that is satisfiable under $\mathcal{O}$ such that $q \subsetneq_{\mathcal{O}} q'$ and $q' \not\subseteq_{\mathcal{O}} q$. Then, there is a homomorphism g from q' to $\mathcal{U}_{q,\mathcal{O}}$ with $g(x_0) = x_0$. We have to show that there is a $p \in F_q$ such that $p \subsetneq_{\mathcal{O}} q'$. To accomplish this, we construct in five steps a homomorphism h from q' to $\mathcal{U}_{p,\mathcal{O}}$ with $h(x_0) = x_0$ for some $p \in F_q$. During the construction, we maintain the invariant

$$h(z)^{\downarrow} = g(z) \tag{$*$}$$

for all variables $z \in \mathsf{var}(q')$ with $h(z)$ defined and $\cdot^{\downarrow}$ the extension of the original $\cdot^{\downarrow}$ for p to a homomorphism from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$. In the first step of the construction, we define h for an initial segment of q'.

Let $U \subseteq \mathsf{var}(q')$ be the smallest set of variables (with regard to $\subseteq$) of q' such that

- $x_0 \in U$, and
- if there is an atom $R(x,y) \in q'$ with $x \in U$ and $S(g(x), g(y)) \in q$ with $\mathcal{O} \models R \equiv S$, then $y \in U$.

Let q^U be the restriction of q' to the variables in U. Note that q^U is connected.

Claim 2. For all $x \in U$ with $q'_x \not\subseteq_{\mathcal{O}} q_{g(x)}$, there is a $p \in F_0(g(x))$ and a homomorphism h_x from q_x^U to $\mathcal{U}_{p,\mathcal{O}}$ that satisfies the invariant $(*)$.

Proof of Claim 2. For readability, set $y = g(x)$. We show Claim 2 by induction on the codepth of x in q^U. In the induction start, x has codepth 0. We distinguish the following cases:

- There is a role atom $R(y, y') \in q_y$.

  Then, let p be the element of $F_0(y)$ that is constructed by generalizing the subquery attached to $R(y, y')$ and define h_x by setting $h_x(x) = y$. Point 2 of generalizing a subquery assures that $y \in A^{\mathcal{U}_{q,\mathcal{O}}}$ implies $y \in A^{\mathcal{U}_{p,\mathcal{O}}}$ for all concept names A. Therefore, h_x is a homomorphism.

- There is no role atom $R(y, y') \in q_y$.

  Then, $q'_x \not\subseteq_{\mathcal{O}} q_y$ implies that there is a concept atom $A(y) \in q_y$ with $x \notin A^{\mathcal{U}_{q'_x,\mathcal{O}}}$, and there must even be a concept name A with these properties and such that there is no $B(y) \in q_y$ with $\mathcal{O} \models B \sqsubseteq A$ and $\mathcal{O} \not\models A \sqsubseteq B$. This implies that Property (a) of dropping concept atoms is satisfied. If Property (b) is not satisfied, then there is a $R(y, y') \in q$ with $\mathcal{O} \models \exists R.\top \sqsubseteq A$. Then, y' must be the predecessor of y, and x cannot be the root of q'. It follows from the definition of U, that there is an $S(x, x') \in q'$ with $\mathcal{O} \models S \equiv R$. Hence, $x \in A^{\mathcal{U}_{q'_x,\mathcal{O}}}$, contradicting that $q'_x \not\subseteq_{\mathcal{O}} q_y$.

  Thus, there is a $p \in F_0(y)$ constructed by dropping the concept atom $A(y)$. Define h_x by setting $h_x(x) = y$.

In the induction step, let $x \in U$ be a variable with codepth > 0 in q^U and assume that the claim holds for all variables of smaller codepth. From $q'_x \not\subseteq_{\mathcal{O}} q_y$ it follows that $q_y(y) \nrightarrow \mathcal{U}_{q'_x,\mathcal{O}}, x$. We distinguish the following cases:

- There is an $R(y, y') \in q_y$ such that $q_{y'}(y') \nrightarrow \mathcal{U}_{q'_{x'},\mathcal{O}}, x'$ for all $S(x, x') \in q'_x$ with $\mathcal{O} \models S \sqsubseteq R$.

  If $\mathsf{func}(R) \in \mathcal{O}$, then $S = R$. If there is an $R(x, x') \in q'_x$, then, by the induction hypothesis, there is a $p' \in F_0(y')$ such that $q'_{x'}(x') \to \mathcal{U}_{p',\mathcal{O}}, x'$. Let $p \in F_0(y)$ be the query constructed by generalizing the subquery attached to $R(y, y')$ and choosing p' in Point 4.

  If $\mathsf{func}(R) \notin \mathcal{O}$, then let $p \in F_0(y)$ be constructed by generalizing the subquery attached to the role atom $R(y, y')$.

  We construct the homomorphism h_x from q_x^U to $\mathcal{U}_{p,\mathcal{O}}$ by starting with $h_x(x) = y$ and continuing to map all successors of x. Let $S(x, x') \in q_x^U$.

If $g(x') \neq y'$, then define h_x for the subtree below x' by setting $h_x(z) = g(z)$ for all $z \in \mathsf{var}(q_{x'})$.

If $g(x') = y'$, then the definition of U implies that $\mathcal{O} \models S \equiv R$. By the induction hypothesis, there is a $p' \in F_0(y')$ and a homomorphism $h_{x'}$ from $q'_{x'}$ to $\mathcal{U}_{p',\mathcal{O}}$ with $h_{x'}(x') = y'$. Extend h_x to the variables in $q'_{x'}$ by setting $h_x(z) = h_{x'}(z)$ for all $z \in \mathsf{var}(q_{x'})$ where $h_{x'}$ is considered to map into the copy of p' that was attached to y in Point 3 or Point 4 of generalizing a subquery.

- For every $R(y, y') \in q_y$ there is a $S(x, x') \in q'_x$ with $q_{y'}(y') \to \mathcal{U}_{q'_{x'},\mathcal{O}}, x'$ and $\mathcal{O} \models S \sqsubseteq R$.

  Then there is an $A(y) \in q_y$ with $x \notin A^{\mathcal{U}_{q'_x,\mathcal{O}}}$ and there must even be an A with these properties such that there is no $B(y) \in q_y$ with $\mathcal{O} \models B \sqsubseteq A$ and $\mathcal{O} \not\models A \sqsubseteq B$. Thus, Property (a) of dropping concept atoms is satisfied. To show that Property (b) is also satisfied, we have to argue that there is no $R(y, y') \in q$ with $\mathcal{O} \models \exists R.\top \sqsubseteq A$. If y' is a successor of y, then $q_{y'}(y') \to \mathcal{U}_{q'_{x'},\mathcal{O}}, x'$ for some $S(x, x') \in q'_x$ with $\mathcal{O} \models S \sqsubseteq R$. This implies $A(x) \in \mathcal{U}_{q'_x,\mathcal{O}}$, a contradiction. Hence, y' must be a predecessor of y. Then x is not the root of q' and by definition of U, there is a predecessor x'' of x with $S(x, x'') \in q'$ and $\mathcal{O} \models R \equiv S$. This implies $A(x) \in \mathcal{U}_{q'_x,\mathcal{O}}$, a contradiction.

  We may thus construct $p \in F_0(y)$ by dropping the concept atom $A(y)$. Set $h_x(z) = g(z)$ for all $z \in \mathsf{var}(q_x^U)$.

This completes the proof of Claim 2.

By Claim 2, there is a $p' \in F_0(x_0)$ such that $q^U(x_0) \to \mathcal{U}_{p',\mathcal{O}}, x_0$. Let $p \in F$ be the query that was obtained by applying Step 2 to p'. Then clearly also $q^U(x_0) \to \mathcal{U}_{p,\mathcal{O}}, x_0$. Define h for all variables in U according to the homomorphism h_{x_0}.

In the second step of the construction of h, we consider atoms $R(x, x') \in q'$ with $h(x)$ defined, $h(x')$ undefined and $S(g(x), g(x')) \in q$ for some role S with $\mathcal{O} \not\models S \equiv R$. Then $\mathcal{O} \models S \sqsubseteq R$. If $h(x)$ is the root of a $p \in F_0(g(x))$ that was created by generalizing the subquery below $S(g(x), g(x'))$, then Step 5 of generalizing a subquery added a disjoint copy of $q_{g(x')}$ and an atom $R(h(x), g(x'))$. Set $h(x') = g(x')$ and continue to map the subtree below x' into the copy of $q_{g(x')}$ according to g until an atom $R'(z, z')$ is encountered with $h(z)$ defined, $h(z')$ undefined and $g(z') \notin \mathsf{var}(q)$ or $S'(g(z), g(z'))$ is directed towards x_0. Otherwise, if $h(x)$ is not the root of a $p \in F_0(g(x))$, p contains an atom $S(h(x), g(x'))$ and the entire subtree $q_{g(x')}$. Set $h(x') = g(x')$ and continue mapping the subtree $q'_{x'}$ as in the previous case.

We continue with the third step of the construction of h which covers subtrees of q' that are connected to the initial segment q^U and whose root is mapped by g to traces of $\mathcal{U}_{q,\mathcal{O}}$ (rather than to a variable from $\mathsf{var}(q)$). Consider all atoms $R(x, x') \in q'$

with $h(x)$ defined, $h(x')$ undefined and $g(x') \notin \mathsf{var}(q)$. Before extending h to $q'_{x'}$, we first show that there is an atom $S(h(x), z) \in p$ with $\mathcal{O} \models S \sqsubseteq R$, added in Step 2 A.

Since $g(x') \notin \mathsf{var}(q)$, $g(x')$ must be a trace $g(x)SM \in \Delta^{\mathcal{U}_{q,\mathcal{O}}}$ for some set of concept names $M = \{A_1, \dots, A_k\}$ and some role S with $\mathcal{O} \models S \sqsubseteq R$. Hence, $g(x) \rightsquigarrow^{S}_{q,\mathcal{O}} M$. We aim to show that Step 2 A of compensation is applicable. To this end, take any concept name B such that $\mathcal{O} \models \exists R.\top \sqsubseteq B$ and $B(g(x)) \in q$. We have to show that $B(h(x)) \in p$. Assume to the contrary that $B(h(x)) \notin p$. Then, p must be the result of dropping the concept atom $B(g(x))$. Since $x \in U$, the choice of p and construction of h in the proof of Claim 2 imply that $x \notin B^{\mathcal{U}_{q'_{x'},\mathcal{O}}}$. However, $R(x, x') \in q'$ implies that $x \in B^{\mathcal{U}_{q'_{x'},\mathcal{O}}}$, a contradiction.

Hence, Step 2 A adds the atoms $R(h(x), z), A_1(z), \dots, A_k(z), S(z', z)$ with z and z' fresh variables and adds a disjoint copy $\widehat{q}$ of q, gluing the copy of $h(x)^{\downarrow}$ in $\widehat{q}$ to z'. Extend h to the variables in $q'_{x'}$ by setting $h(\widehat{x}) = zR_2M_2 \dots R_nM_n$ if $g(\widehat{x}) = g(x)SMR_2M_2 \dots R_nM_n$ for all $\widehat{x}$ in the subtree below x'. If there is an $x'' \in \mathsf{var}(q'_{x'})$ with $g(x'') = g(x)$, then instead set $h(x'') = z'$ and continue mapping the subtree below x'' into the attached copy $\widehat{q}$ of q according to g.

In the fourth step of the construction of h, we consider the remaining subtrees of q'. Let $R(x, x') \in q'$ be directed away from x_0 with $h(x)$ defined and $h(x')$ undefined.

Then $h(x)$ was defined in the first step of the construction of h, and thus $x \in U$. As $h(x')$ was not defined in the first or second step, $x' \notin U$ and $g(x') \in \mathsf{var}(q)$. Therefore, $R(g(x), g(x')) \in \mathcal{U}_{q,\mathcal{O}}$ must be directed towards x_0. This implies that x is not the root of q' and that there is an atom $T(x, x'') \in q'$ directed towards x_0 with $g(x'') = g(x')$. From $x \in U$ follows that $x'' \in U$ and therefore $h(x'')$ and $h(x)$ were defined in the first step of the construction of h.

Since h is a homomorphism where it is defined, there is an atom $S(h(x), h(x'')) \in p$ directed towards x_0 with $\mathcal{O} \models S \sqsubseteq T$ that was not added in Step 2 A. By the invariant (*), $h(x'')^{\downarrow} = g(x'') = g(x')$ and $h(x)^{\downarrow} = g(x)$, therefore $\mathcal{A}_q, \mathcal{O} \models R(h(x)^{\downarrow}, h(x'')^{\downarrow})$ and $\mathsf{func}(R) \notin \mathcal{O}$.

Therefore, the start of Step 2 B added an atom $R(h(x), z)$ to p where z is a fresh variable, and marked it. Set $h(x') = z$. We continue to map the subtree $q'_{x'}$ in the next step of the construction of h.

In the fifth and final step of the construction of h we define h for all remaining variables using the atoms that were introduced in the iteration of Step 2 B. We do this by repeatedly choosing atoms $R(x, x') \in q'$ directed away from x_0 such that

1. $h(x)$ and $h(x')$ are defined, and

2. for all $S(x', x'') \in q'$ directed away from x_0, $h(x'')$ is undefined and there is at least one such $S(x', x'')$.

If we choose such an $R(x, x')$ directly at the beginning of this step of the construction of h, then $g(x') \in \mathsf{var}(q)$ and there is an atom $R'(h(x), h(x')) \in p$ with $\mathcal{O} \models R' \sqsubseteq R$ that was marked and processed in Step 2 B of the construction of p. We will extend h such that these conditions are always satisfied when we choose an $R(x, x') \in q'$.

Let $R(x, x') \in q'$ be an atom that satisfied Properties 1 and 2 and consider the atom $R'(h(x), h(x')) \in p$. If $\mathsf{func}(R'^{-}) \notin \mathcal{O}$, then Step 2 B attached a copy of q to $h(x')$. Extend h to the entire subtree $q'_{x'}$ by setting $h(z) = g(z)$ for all $z \in \mathsf{var}(q'_{x'})$ where g is considered to be a homomorphism into that copy of q. If $\mathsf{func}(R'^{-}) \in \mathcal{O}$, then consider each $S(x', x'') \in q'$ directed away from x_0. We distinguish cases

- $g(x'') \in \mathsf{var}(q)$. Since $g(x') \in \mathsf{var}(q)$, there is an atom $S'(g(x'), g(x'')) \in q$ for some role name S' with $\mathcal{O} \models S' \sqsubseteq S$. If $S = R'^{-}$, then $\mathsf{func}(S) \in \mathcal{O}$ and $S = S'$. Additionally, since $\mathsf{func}(R'^{-}) \in \mathcal{O}$, there must be the atom $R'(g(x), g(x')) = S^{-}(g(x), g(x')) \in q$. This contradicts that q satisfies all functionality constraints in $\mathcal{O}$, implying that $S \neq R'^{-}$.

  Therefore, Point 2 of Step 2 B adds the atom $S(h(x'), z)$ where z is a fresh variable. Set $h(x'') = z$.

- $g(x'') \notin \mathsf{var}(q)$. Since $g(x') \in \mathsf{var}(q)$, $g(x'')$ must be of the shape $g(x')S'M$ for some role S' with $\mathcal{O} \models S' \sqsubseteq S$ and some set $M = \{A_1, \dots A_k\}$ of concept names. Hence, $g(x') \rightsquigarrow^{S'}_{q,\mathcal{O}} M$ and since $h(x')^{\downarrow} = g(x')$, Point 3 of Step 2 B added the atoms $S'(h(x'), u), A_1(u), \dots A_k(u), S'^{-}(u, y')$ where u and y' are fresh variables. Set $h(x'') = u$ and extend h to the initial segment of $q'_{x''}$ that maps into the traces below $g(x')S'M$ by setting $h(z) = uR_2M_2t$ if $g(z) = g(x')S'MR_2M_2t$ for all $z \in \mathsf{var}(q'_{x''})$ until $g(z) = g(x')$. If an atom $T(z, z') \in q_{x''}$ with $h(z)$ defined and $g(z') = g(x')$ is encountered, set $h(z') = y'$. As $T(z, z')$ fulfills Properties 1 and 2, we will extend h to its subtree at some point in the future.

This completes the construction of h and the proof that the second condition of frontiers is satisfied. □

We next show that the constructed frontier is of polynomial size and that its computation takes only polynomial time.

Lemma 4.27. *The construction of F_q runs in time polynomial in $\|q\| + \|\mathcal{O}\|$ and $\sum_{p \in F_q} \|p\|$ is polynomial in $\|q\| + \|\mathcal{O}\|$.*

Proof. In order to reduce notational clutter, we introduce some abbreviations used throughout the proof.

- $s = |\mathsf{sig}(q)|$ denotes the number of concept and role names used in q;
- $o = \|\mathcal{O}\|$ denotes the size of $\mathcal{O}$;

- for an ELIQ p, $n_p = |\mathsf{var}(p)|$ denotes the number of variables in p;
- for a set Q of ELIQs, n_Q denotes $\sum_{p \in Q} n_p$.

We assume without loss of generality that s and o are at least one.

We start with analyzing the size of the queries in $F_0(x)$ that are obtained as the result of the generalization step.

Claim. For every $x \in \mathsf{var}(q)$,

1. $|F_0(x)| \leq s \cdot n_{q_x}$, and
2. $n_{F_0(x)} \leq s \cdot o \cdot n_{q_x}^3$.

Proof of the claim. The proof of both points is by induction on the codepth of x in q. We start with Point 1. For the base case, consider a variable x of codepth 0 in q, that is, a leaf. In this case, only *Drop a concept atom* is applicable, and the construction adds at most s queries to $F_0(x)$.

For the inductive step, consider a variable x of codepth greater than 0. We partition $F_0(x)$ into $F_0^A(x)$ and $F_0^B(x)$, that is, the queries that are obtained by dropping a concept atom and the queries that are obtained by generalizing a subquery, respectively, and analyze them separately, starting with $F_0^A(x)$. Clearly there are at most s queries in $F_0^A(x)$, that is

$$|F_0^A(x)| \leq s.$$

Next, we analyze $F_0^B(x)$. Each query in $F_0^B(x)$ is obtained by picking, in Point 1, an atom $R(x, y) \in q_x$. If $\mathsf{func}(R) \notin \mathcal{O}$, then Point 3 adds one query to $F_0^B(x)$. Otherwise, Point 4 adds $|F_0(y)|$ queries. Thus,

$$|F_0^B(x)| \leq \sum_{\substack{R(x,y) \in q_x \\ \mathsf{func}(R) \notin \mathcal{O}}} 1 + \sum_{\substack{R(x,y) \in q_x \\ \mathsf{func}(R) \in \mathcal{O}}} |F_0(y)|.$$

Using the induction hypothesis and the fact that $n_{q_y} \geq 1$, we obtain

$$|F_0^B(x)| \leq \sum_{R(x,y) \in q_x} s \cdot n_{q_y} = s \cdot \sum_{R(x,y) \in q_x} n_{q_y} = s \cdot (n_{q_x} - 1).$$

Hence,

$$|F_0(x)| = |F_0^A(x)| + |F_0^B(x)| \leq s + s \cdot (n_{q_x} - 1) = s \cdot n_{q_x}.$$

We now prove Point 2 by induction on the codepth of x in q. For the base case, consider a variable x of codepth 0 in q, that is, a leaf. In this case, only *Drop a concept atom* is applicable, and it adds at most s queries to $F_0(x)$, each with a single variable.

For the inductive step, consider a variable x of codepth greater than 0 and the same partition of $F_0(x)$ into $F_0^A(x)$ and $F_0^B(x)$ as before. Every $p \in F_0^A(x)$ uses n_{q_x} variables, and there are at most s queries in $F_0^A(x)$. Thus,

$$n_{F_0^A(x)} \leq s \cdot n_{q_x}.$$

Next, we analyze $F_0^B(x)$. Each query in $F_0^B(x)$ is obtained by first picking, in Point 1, an atom $R(x,y)$ in q_x. If $\mathsf{func}(R) \notin \mathcal{O}$, Point 3 adds $n_{F_0(y)}$ variables. Otherwise, Point 4 replaces q_y with some element of $F_0(y)$. Then, Point 5 adds some copies of q_y, depending on the number of role inclusions in $\mathcal{O}$. Hence,

$$\begin{aligned} n_{F_0^B(x)} &\leq \sum_{\substack{R(x,y)\in q_x \\ \mathsf{func}(R)\notin\mathcal{O}}} (n_{q_x} + n_{F_0(y)} + o \cdot n_{q_y}) + \sum_{\substack{R(x,y)\in q_x \\ \mathsf{func}(R)\in\mathcal{O}}} (n_{q_x} \cdot |F_0(y)| + n_{F_0(y)} + o \cdot n_{q_y} \cdot |F_0(y)|) \\ &\leq \sum_{R(x,y)\in q_x} (n_{q_x} \cdot |F_0(y)| + n_{F_0(y)} + o \cdot n_{q_y} \cdot |F_0(y)|). \end{aligned}$$

Plugging in the induction hypothesis and Point 1 of the claim, we obtain

$$\begin{aligned} n_{F_0^B(x)} &\leq \sum_{R(x,y)\in q_x} (n_{q_x} \cdot s \cdot n_{q_y} + s \cdot o \cdot n_{q_y}^3 + o \cdot n_{q_y} \cdot s \cdot n_{q_y}) \\ &= s \cdot n_{q_x} \cdot \sum_{R(x,y)\in q_x} n_{q_y} + s \cdot o \cdot \sum_{R(x,y)\in q_x} n_{q_y}^3 + s \cdot o \cdot \sum_{R(x,y)\in q_x} n_{q_y}^2. \end{aligned}$$

We simplify the right-hand side by observing that

$$\sum_{R(x,y)\in q_x} n_{q_y} = n_{q_x} - 1$$

and for $k \geq 0$

$$\sum_{R(x,y)\in q_x} n_{q_y}^k \leq \left(\sum_{R(x,y)\in q_x} n_{q_y} \right)^k = (n_{q_x} - 1)^k.$$

Here, the inequality is an application of the general inequality $\sum_i a_i^k \leq \left(\sum_i a_i\right)^k$, for every sequence of non-negative numbers $a_1, \ldots, a_m$ and $k \geq 1$. Using these observations, the inequality can be simplified to:

$$\begin{aligned} n_{F_0^B(x)} &\leq s \cdot n_{q_x} \cdot (n_{q_x} - 1) + s \cdot o \cdot (n_{q_x} - 1)^3 + s \cdot o \cdot (n_{q_x} - 1)^2 \\ &\leq s \cdot o \cdot \left(n_{q_x} \cdot (n_{q_x} - 1) + (n_{q_x} - 1)^3 + (n_{q_x} - 1)^2 \right) \\ &= s \cdot o \cdot \left((n_{q_x} - 1)^3 + 2n_{q_x}^2 - 3n_{q_x} + 1 \right). \end{aligned}$$

Overall, we get

$$\begin{aligned} n_{F_0(x)} &= n_{F_0^A(x)} + n_{F_0^B(x)} \\ &\leq s \cdot n_{q_x} + s \cdot o \cdot \left((n_{q_x} - 1)^3 + 2n_{q_x}^2 - 3n_{q_x} + 1 \right) \\ &\leq s \cdot o \cdot \left((n_{q_x} - 1)^3 + 2n_{q_x}^2 - 2n_{q_x} + 1 \right) \\ &\leq s \cdot o \cdot n_{q_x}^3. \end{aligned}$$

For the last inequality, we used that $z^3 \geq (z-1)^3 + 2z + 1$, for all real numbers z. This finishes the proof of the claim.

We now analyze the Step 2, in which the queries in $F_0(x_0)$ are further extended. We denote with F_1 the result of applying Step 2 A to $F_0(x_0)$. In Step 2 A, we add at most one variable and a copy of q for every variable in $F_0(x_0)$ and concept $\exists R.B$ or role inclusion $R \sqsubseteq S$ in $\mathcal{O}$.

Therefore, the step adds at most $(1 + n_q) \cdot n_{F_0(x_0)} \cdot o$ variables in total. Using the bound on $n_{F_0(x_0)}$, we get

$$\begin{aligned} n_{F_1} &\leq n_{F_0(x_0)} + (1 + n_q) \cdot n_{F_0(x_0)} \cdot o \\ &\leq s \cdot o \cdot n_q^3 \cdot \left(1 + (1 + n_q) \cdot o \right). \end{aligned}$$

We now analyze Step 2 B, applied to some query $p \in F$. We argue that the iteration terminates after a polynomial number of steps, thus resulting in a query of polynomial size.

Consider an atom $R(x, y)$ that was marked. If $\mathsf{func}(R^-) \notin \mathcal{O}$, then a copy of q is attached and no new atoms are marked. Otherwise, for all atoms $S(y^{\downarrow}, z) \in q$ and roles S' with $\mathcal{O} \models S \sqsubseteq S'$ such that $S' \neq R^-$, new atoms are added and marked. For all new atoms with $S \neq S'$ no new atoms are marked when they are processed, as $\mathcal{O} \models S^- \sqsubseteq S'^-$ and hence $\mathsf{func}(S'^-) \notin \mathcal{O}$. All new atoms with $S = S'$ must be copies of atoms in q, and, due to the $S' \neq R^-$ condition and the tree-shape of q, the marking process never changes its direction and creates at most a single copy of each atom in q. Overall, we obtain that, per role atom in p, the marking process adds at most $n_q \cdot o$ role atoms in Step 2, for each such atom and every $\exists R.B$ in $\mathcal{O}$ one more role atom in Step 3, and for each introduced variable at most one copy of q. All this is polynomial in $\|q\|$ and $\|\mathcal{O}\|$.

Moreover, the computation of F_q can be carried out in polynomial time since all involved queries are of polynomial size and consequences of $\mathcal{O}$ can be decided in polynomial time. □

From Lemma 4.26 and Lemma 4.27 it now follows that the frontier construction in this section actually yields frontiers of ELIQs under $\textit{DL-Lite}_{\text{core}}^{\mathcal{HF}^-}$ ontologies in polynomial time. Thus, we have shown Theorem 4.23.

Lower Bounds on the Size of Frontiers

The construction of a frontier in polynomial time crucially relies on the choice of ontology language. In the remainder of this section we show that any extension of the ontology language *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ by including conjunctions, unrestricted functionality constraints, or qualified existential restrictions, leads to frontiers of exponential or even infinite size. These results are interesting on their own, as they already hold for simple query and ontology languages.

In a sense, the first two results mirror the learning lower bounds in Section 4.1. Indeed, we can view the non-existence of frontiers of polynomial size as the reason for the impossibility of polynomial query learning. Again, the first result concerns conjunctions in the ontology language, and uses the same queries and ontologies as the proof of Theorem 4.5.

Theorem 4.28. *For every $n \geq 1$, there is a conjunction of atomic queries q_n and a conjunctive ontology $\mathcal{O}_n$ of size polynomial in n, such that any frontier of q_n under $\mathcal{O}_n$ has size at least 2^n.*

Proof. For $n \geq 1$, let $A_1, \dots, A_n, B_1, \dots, B_n$ be concept names and let

$$q_n(x) \leftarrow A_1(x) \land B_1(x) \land \cdots \land A_n(x) \land B_n(x),$$
$$\mathcal{O}_n = \{A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n \mid 1 \leq i \leq n\}.$$

Suppose a set of queries F is a frontier of q_n under $\mathcal{O}_n$. Let p be any query that for each i with $1 \leq i \leq n$ contains either $A_i(x)$ or $B_i(x)$. It suffices to show that $p \in F$.

Clearly, $q_n \subsetneq_{\mathcal{O}_n} p$ and $p \not\subseteq_{\mathcal{O}_n} q_n$. Hence, the second condition of Definition 4.13 implies that there is a $p' \in F$ with $p' \subseteq_{\mathcal{O}} p$. Since $p' \in F$, it must be that $p' \not\subseteq_{\mathcal{O}_n} q_n$, and therefore p' does not contain both atoms $A_i(x), B_i(x)$ for any i.

But then the ontology does not have an effect on $p' \subseteq_{\mathcal{O}_n} p$ and hence every atom that occurs in p must occur in p'. As p' does not contain both atoms $A_i(x), B_i(x)$ for any i, it follows that $p' = p$, which was to be shown. □

If we lift the restriction on the interaction of functionality constraints and existential restrictions, so consider proper *DL-Lite*$_{\text{core}}^{\mathcal{F}}$ ontologies, Theorem 4.23 also fails, even if we permit frontiers that consist of CQs. A *CQ-frontier* of an ELIQ q under an ontology $\mathcal{O}$ is a finite set of unary CQs that satisfies Conditions 1 and 2 of Definition 4.13. Note that every frontier is a CQ-frontier, but not vice versa.

Theorem 4.29. *There is an ELIQ q and a DL-Lite*$_{\text{core}}^{\mathcal{F}}$ *ontology $\mathcal{O}$ such that every CQ-frontier of q under $\mathcal{O}$ is infinite.*

Proof. Let $q(x) \leftarrow A(x)$ and

$$\mathcal{O} = \{\, A \sqsubseteq \exists r.\top, \quad \exists r^-.\top \sqsubseteq \exists r.\top, \quad \exists r.\top \sqsubseteq \exists s.\top, \quad \mathsf{func}(r^-) \,\}.$$

The universal model $\mathcal{U}_{q,\mathcal{O}}$ of $\mathcal{A}_q$ and $\mathcal{O}$ is an infinite r-path in which every point has a single s-successor.

Suppose, for the sake of showing a contradiction, that F is a CQ-frontier of q under $\mathcal{O}$. We can assume without loss of generality that all queries in F are satisfiable under $\mathcal{O}$, especially that they satisfy $\mathsf{func}(r^-)$. Since F is finite, there is an $n \geq 1$ such that $|\mathsf{var}(p)| < n$, for all $p \in F$. Consider the following ELIQ q':

$$\begin{aligned} q'(x_1) \leftarrow\ & r(x_1, x_2) \wedge \cdots \wedge r(x_{n-1}, x_n) \wedge \\ & s(x_n, y) \wedge s(x'_n, y) \wedge \\ & r(x'_1, x'_2) \wedge \cdots \wedge r(x'_{n-1}, x'_n) \wedge A(x'_1). \end{aligned}$$

Note that $q' \not\subseteq_{\mathcal{O}} q \subseteq_{\mathcal{O}} q'$ and that q' satisfies $\mathsf{func}(r^-)$.

By the second condition of frontiers, there is a query $p(z) \in F$ such that $p \subseteq_{\mathcal{O}} q'$. By Lemma 3.5, there is a homomorphism h from q' to $\mathcal{U}_{p,\mathcal{O}}$ with $h(x_1) = z$. We distinguish cases.

Suppose first that $h(x_i) \in \mathsf{var}(p)$ for all i with $1 \leq i \leq n$, then by the choice of n there must be $1 \leq i < j \leq n$ such that $h(x_i) = h(x_j)$. Since q' contains a directed r-path from x_i to x_j and $\mathcal{U}_{p,\mathcal{O}}$ does not contain edges between variables that are not part of p, this implies that p must contain an r-cycle. Thus, $q \not\subseteq_{\mathcal{O}} p$, violating the first condition of frontiers.

Suppose now that $h(x_i) \notin \mathsf{var}(p)$ for some i with $1 \leq i \leq n$, that is, $h(x_i)$ is a trace starting with some $y \in \mathsf{var}(p)$. Since q' is an ELIQ, there is a $j < i$ such that $h(x_j) = y$ and $h(x_{j+1}), \ldots, h(x_i) \notin \mathsf{var}(p)$. The structure of q' and the structure of the proper traces in universal models of $\mathcal{O}$ imply that $h(x'_j) = h(x_j)$.

We now show that $h(x_1) = h(x'_1)$. If $j = 1$, we are done. If $j > 1$, there are atoms $r(x_{j-1}, x_j)$ and $r(x'_{j-1}, x'_j)$ in q'. Since h is a homomorphism, $h(x_j) = h(x'_j)$, and p satisfies $\mathsf{func}(r^-)$, we obtain $h(x_{j-1}) = h(x'_{j-1})$. Repeating this argument yields $h(x_1) = h(x'_1)$ as required. Since $h(x_1) = z$, we also have $h(x'_1) = z$. Since h is a homomorphism and $A(x'_1) \in q'$, we have $A(z) \in p$ and thus $p \subseteq_{\mathcal{O}} q$, violating the first condition of frontiers. □

Theorem 4.28 already implies that there do not always exist frontiers of polynomial size under $\mathcal{EL}$ ontologies. Kriegel showed that there even are cases where no finite frontiers[4] exist under an $\mathcal{EL}$ ontology that does not use any conjunction [Kri18a]. We give a self-contained proof of Theorem 4.30 using the terminology of this section.

Theorem 4.30 ([Kri18a]). *There is an ELQ q and an $\mathcal{EL}$ ontology $\mathcal{O}$ that does not contain any conjunctions, such that every frontier of q under $\mathcal{O}$ is infinite.*

[4] Upper neighborhoods of $\mathcal{EL}$ concepts correspond to minimal finite frontiers.

Proof. Let $\mathcal{O} = \{A \equiv \exists r.A\}$ and $q(x) \leftarrow A(x)$. Suppose, for the sake of showing a contradiction, that a finite set of ELIQs F is a frontier of q under $\mathcal{O}$.

Consider, for each $i \geq 1$, the ELQ

$$p_i(x_0) \leftarrow r(x_0, x_1) \wedge \cdots \wedge r(x_{i-1}, x_i).$$

For all $i \geq 1$, $q \subseteq_{\mathcal{O}} p_i$ and $p_i \not\subseteq_{\mathcal{O}} q$. Hence, by definition of frontiers, for each i there is a $p \in F$ such that $p \subseteq_{\mathcal{O}} p_i$. As F is finite, there must be a $p \in F$ such that $p \subseteq_{\mathcal{O}} p_i$ for infinitely many i. We distinguish cases.

If the concept name A occurs in p, then $p \equiv_{\mathcal{O}} q$, contradicting that F is a frontier.

If only the role name r occurs in p, then $\Delta^{\mathcal{U}_{p,\mathcal{O}}} = \mathsf{var}(p)$. As p itself is also finite, $\mathcal{U}_{p,\mathcal{O}}$ cannot contain an r-path of infinite length. Hence, it cannot be true that $p \subseteq_{\mathcal{O}} p_i$ for infinitely many i. □

Recall that *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ ontologies also restrict the interaction of functionality constraints and role inclusions, that is, no functional role may have subroles. We used this restriction heavily in our proof of Theorem 4.23. We show that this restriction is indeed essential for (finite) frontiers to exist. We conjecture that this restriction is also necessary for polynomial time learning of queries using only membership queries.

Theorem 4.31. *There is an ELQ q and an ontology $\mathcal{O}$ consisting only of role inclusions and functionality constraints, such that every frontier of q under $\mathcal{O}$ is infinite.*

Proof. Let $q(x_0) \leftarrow r(x_0, x_1)$ and

$$\mathcal{O} = \{r \sqsubseteq s, r \sqsubseteq t, \mathsf{func}(s), \mathsf{func}(t), \mathsf{func}(s^-), \mathsf{func}(t^-)\},$$

and assume for contradiction that F is a finite frontier of q under $\mathcal{O}$. For every $i \geq 1$, consider the query

$$q_i(y_0) \leftarrow s(y_0, y_0') \wedge t(y_1, y_0') \wedge \cdots \wedge s(y_{i-1}, y_{i-1}') \wedge t(y_i, y_{i-1}').$$

Every q_i is satisfiable under $\mathcal{O}$, and it holds that $q \subseteq_{\mathcal{O}} q_i$ and $q_i \not\subseteq_{\mathcal{O}} q$.

Therefore, for each q_i there must be an ELIQ $p \in F$ such that $p \subseteq_{\mathcal{O}} q_i$. Since F is finite, there must be a $p(z_0) \in F$ such that $p \subseteq_{\mathcal{O}} q_i$ for infinitely many q_i. As F is a frontier, $p \not\subseteq_{\mathcal{O}} q$. Hence, this p may not contain an r atom at the root. Due to the functionality constraints in $\mathcal{O}$, p is only satisfiable under $\mathcal{O}$ if it does not contain an r atom at all. Now consider a q_i with $p \subseteq_{\mathcal{O}} q_i$ and $i > |\mathsf{var}(q_f)|$. Since p contains no r atoms, also $p \subseteq_{\varnothing} q_i$. Let h be a homomorphism from q_i to p with $h(y_0) = z_0$. It follows from $|\mathsf{var}(q_i)| > |\mathsf{var}(p)|$ that h is non-injective. This, together with the construction of q_i implies that p is cyclic, contradicting that p is an ELIQ. □

Theorems 4.28 to 4.31 together indicate that *DL-Lite*$_{\text{core}}^{\mathcal{HF}^-}$ is a maximal ontology language for which polynomial size frontiers of ELIQs exist, in the sense that this property does not hold for many of its common extensions.

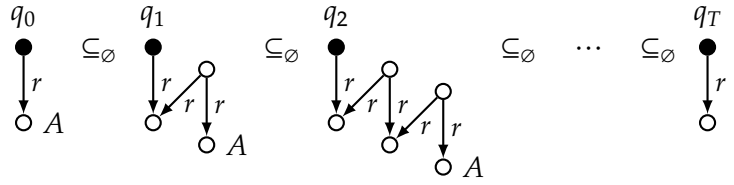

Figure 4.8: An infinite generalizing chain of ELIQs under the empty ontology.

4.4 Generalization Sequences of Queries

We intend to use the frontier construction from Section 4.3 as part of a learning algorithm to approach the target query q_T step-by-step under an $\mathit{DL\text{-}Lite}^{\mathcal{HF}^-}_{\text{core}}$ ontology. In Example 4.24 we can observe that the queries in a frontier of q are usually much larger than q, and the proof of Lemma 4.27 indeed gives an upper bound on their size that is cubic in $\|q\|$. Thus, if the algorithm naively applies the frontier construction multiple times, it does approach the target query, but it produces larger and larger queries, and the algorithm cannot possibly run in polynomial time.

A related issue is the question of how often we need to apply the frontier construction to reach q_T. Already in the case without ontologies, there are infinite chains of generalizing queries. Consider the empty ontology and the ELIQs $q_0(x_0) \leftarrow r(x_0, x_1) \land A(x_1)$ and $q_T(x_0) \leftarrow r(x_0, x_1)$. Then, there is an infinite sequence of ELIQs consisting of

$$q_i(x_0) \leftarrow r(x_0, x_1) \land r(x_2, x_1) \land \cdots \land r(x_{2i}, x_{2i-1}) \land r(x_{2i}, x_{2i+1}) \land A(x_{2i+1})$$

for all $i \geq 1$, with $q_i \subseteq_\emptyset q_{i+1}$, $q_{i+1} \not\subseteq_\emptyset q_i$, and $q_i \subseteq_\emptyset q_T$ for all $i \geq 0$. Additionally, it can be verified that q_{i+1} must occur in the frontier of q_i for all $i \geq 0$. The first steps of this sequence are displayed in Figure 4.8. This indicates that if we naively apply the frontier construction, we might end up following an infinite chain of ELIQs that increase in size, and never reach q_T.

A solution to this problem lies in the observation that while q_1 is larger than q_0, not all atoms of q_1 are necessary for $q_1 \subseteq_\emptyset q_T$ to hold. Indeed, if we remove all atoms that mention the variables x_2 and x_3 from q_1, then q_1 becomes $(q_T, \mathcal{O})$-minimal and $q_1 \equiv_\emptyset q_T$. When a learning algorithm selects a new query p from the frontier of the current hypothesis q, the algorithm, of course, cannot inspect q_T to safely remove parts of p to obtain $(q_T, \mathcal{O})$-minimality. However, a learning algorithm can check if the removal of some atoms is safe by using membership queries to verify that $p \subseteq_\mathcal{O} q_T$ holds still.

For this, we define a subroutine $\mathsf{minimize}_\mathcal{O}$ that we will use as part of our learning algorithm. Again, we define the subroutine for all CQs, not just ELIQs and for

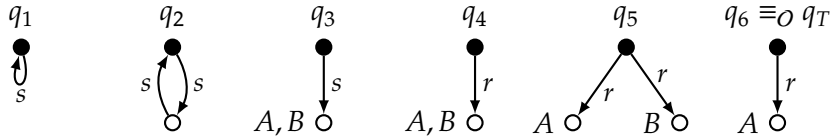

Figure 4.9: The unary rooted CQs $q_1, \dots, q_6$ form a generalization sequence towards $q_T(x_0) \leftarrow r(x_0, x_1) \wedge A(x_1)$ under $\mathcal{O} = \{\exists s \sqsubseteq A \sqcap B,\ s \sqsubseteq r\}$.

multiple ontology languages, since we will reuse $\mathsf{minimize}_{\mathcal{O}}$ in later sections. Let $\mathcal{O}$ be an $\mathcal{EL}^r$, *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ or *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontology, and assume that membership queries are answered with regard to $\mathcal{O}$ and a target CQ q_T. The subroutine $\mathsf{minimize}_{\mathcal{O}}$ takes as input a CQ q that is satisfiable under $\mathcal{O}$ such that $q \subseteq_{\mathcal{O}} q_T$ and computes a $(q_T, \mathcal{O})$-minimal CQ q' such that $q \subseteq_{\mathcal{O}} q'$ and $q' \subseteq_{\mathcal{O}} q_T$. It does this by applying the following operation exhaustively:

Drop variable. Select an existential variable $x \in \mathsf{var}(q)$. Use a membership query to test whether $q^{-o^x} \subseteq_{\mathcal{O}} q_T$. If yes, continue with $q^{-\mathcal{O}x}$ instead of q, otherwise continue with q.

Using Definition 4.18, it is easy to see that $\mathsf{minimize}_{\mathcal{O}}$ achieves $(q_T, \mathcal{O})$-minimality.

Lemma 4.32. *Let q be a CQ with $q \subseteq_{\mathcal{O}} q_T$ that is satisfiable under $\mathcal{O}$. Then, $\mathsf{minimize}_{\mathcal{O}}(q)$ terminates in time polynomial in $\|\mathcal{O}\| + \|q\| + \|q_T\|$ and returns a $(q_T, \mathcal{O})$-minimal CQ q' such that $q \subseteq_{\mathcal{O}} q' \subseteq_{\mathcal{O}} q_T$.*

Note that Lemma 4.32 does not hold if $\mathcal{O}$ is an $\mathcal{ELI}$ or $\mathcal{ELIHF}_\bot$ ontology, as then q^{-o^x} cannot be computed in polynomial time as deciding whether $\mathcal{A}_q, \mathcal{O} \models A(y)$ for some concept name A is ExpTime-complete.

Next, we show that applying $\mathsf{minimize}_{\mathcal{O}}$ to obtain $(q_T, \mathcal{O})$-minimality of hypotheses is sufficient to guarantee that the learning algorithm reaches q_T after a polynomial number of applications of the frontier construction, and that the involved queries stay bounded in size. First, we formalize the notion of approaching q_T by generalizing hypotheses.

Definition 4.33 (Generalization Sequence)**.** Let q_T be a CQ and $\mathcal{O}$ an ontology. A sequence $q_1, q_2, \dots$ of CQs is a *generalization sequence towards q_T under $\mathcal{O}$* if for all $i \geq 0$, $q_i \subseteq_{\mathcal{O}} q_{i+1}$, $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$, and $q_i \subseteq_{\mathcal{O}} q_T$.

Example 4.34. Consider the queries $q_1, \dots, q_6$ displayed in Figure 4.9 and the ontology

$$\mathcal{O} = \{\exists s.\top \sqsubseteq A \sqcap B,\ s \sqsubseteq r\}.$$

The queries $q_1, \dots, q_6$ form a generalization sequence towards q_T under $\mathcal{O}$. Note that q_5 is not $(q_T, \mathcal{O})$-minimal.

Then, we are interested in generalization sequences that consist of $(q_T, \mathcal{O})$-minimal CQs. If q_T is rooted, then the length of such a sequence must be bounded by a polynomial. Again, we show this result in a very general way, for rooted CQs and for $\mathcal{ELIHF}_\bot$ ontologies, which include all *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies, as we will also apply it in Chapter 5. Moreover, it is also interesting on its own that such a bound exists even for relatively expressive $\mathcal{ELIHF}_\bot$ ontologies.

Theorem 4.35. *Let q_T be a rooted CQ and $\mathcal{O}$ an $\mathcal{ELIHF}_\bot$ ontology in normal form, and let $q_1, q_2, \ldots$ be a generalization sequence towards q_T under $\mathcal{O}$. If all q_i are $(q_T, \mathcal{O})$-minimal and satisfiable under $\mathcal{O}$, then the sequence has length at most $|\mathsf{var}(q_T)|^3 \cdot (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$.*

Proof. Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology in normal form and q_T a rooted CQ. Further, let $q_1(\overline{x}_1), q_2(\overline{x}_2) \ldots$ be a generalization sequence towards $q_T(\overline{x})$ under $\mathcal{O}$ such that all q_i are $(q_T, \mathcal{O})$-minimal and satisfiable under $\mathcal{O}$. We start by showing that all q_i are rooted and have at most as many variables as q_T.

Claim 1. For all i with $i \geq 1$, $|\mathsf{var}(q_i)| \leq |\mathsf{var}(q_T)|$ and q_i is rooted.

Proof of Claim 1. Since $q_i \subseteq_{\mathcal{O}} q_T$, there is a homomorphism h from q_T to $\mathcal{U}_{q_i,\mathcal{O}}$ with $h(\overline{x}) = \overline{x}_i$. Lemma 4.20 then implies that $\mathsf{var}(q_i) \subseteq \mathsf{img}(h)$. Therefore, $|\mathsf{var}(q_i)| \leq |\mathsf{var}(q_T)|$. Additionally, rootedness of q_i follows from rootedness of q_T.

We show next that the queries q_i have a non-decreasing number of role atoms. Since $q_{i-1} \subseteq_{\mathcal{O}} q_i$, for all $i \geq 2$, we fix homomorphisms h_{i-1} from q_i to $\mathcal{U}_{q_{i-1},\mathcal{O}}$ with $h_{i-1}(\overline{x}_i) = \overline{x}_{i-1}$.

Claim 2. For all $i \geq 2$, $\mathsf{var}(q_{i-1}) \subseteq \mathsf{img}(h_{i-1})$ and $|\mathsf{var}(q_{i-1})| \leq |\mathsf{var}(q_i)|$.

Proof of Claim 2. Since $\mathsf{var}(q_{i-1}) \subseteq \mathsf{img}(h_{i-1})$ implies $|\mathsf{var}(q_{i-1})| \leq |\mathsf{var}(q_i)|$, it suffices to show the former. Assume to the contrary that there is an $x \in \mathsf{var}(q_{i-1})$ with $x \notin \mathsf{img}(h_{i-1})$.

Let $q'_{i-1} = q_{i-1}^{-Ox}$ Then, by Lemma 4.19, h_{i-1} is also a homomorphism from q_i to $\mathcal{U}_{q'_{i-1}}$ with $h_{i-1}(\overline{x}_i) = \overline{x}_{i-1}$. By Lemma 3.8, there is also a homomorphism h from $\mathcal{U}_{q_i,\mathcal{O}}$ to $\mathcal{U}_{q'_{i-1}}$ with $g(\overline{x}_i) = \overline{x}_{i-1}$. Composing h with a homomorphism g from q_T to $\mathcal{U}_{q_i,\mathcal{O}}$ with $g(\overline{x}) = \overline{x}_i$ yields a homomorphism g' from q_T to $\mathcal{U}_{q'_{i-1},\mathcal{O}}$ with $g'(\overline{x}) = \overline{x}_{i-1}$. Therefore, $q'_{i-1} = q_{i-1}^{-Ox} \subseteq_{\mathcal{O}} q_T$, contradicting $(q_T, \mathcal{O})$-minimality of q_{i-1}.

This completes the proof of Claim 2.

Now, we use the two claims to show that the generalization sequence must be finite and that its length is bounded by $|\mathsf{var}(q_T)|^3 \cdot (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$. Claim 2 implies that $|\mathsf{var}(q_{i-1})| \leq |\mathsf{var}(q_i)|$ for all $i \geq 2$. By Claim 1, it suffices to show that the length of any subsequence $q_j, \ldots, q_k$ with $|\mathsf{var}(q_j)| = \cdots = |\mathsf{var}(q_k)|$ is bounded by $|\mathsf{var}(q_T)|^2 \cdot (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$.

Consider any $i \in \{j, \dots, k-1\}$. By Claim 2, $\mathsf{var}(q_i) \subseteq \mathsf{img}(h_i)$, and since $|\mathsf{var}(q_{i+1})| = |\mathsf{var}(q_i)|$, the homomorphism h_i is a bijection between $\mathsf{var}(q_{i+1})$ and $\mathsf{var}(q_i)$. Additionally, it follows from h_i being a homomorphism that

1. for every concept name A and variable $x_1 \in \mathsf{var}(q_{i+1})$, $x_1 \in A^{\mathcal{U}_{q_{i+1},\mathcal{O}}}$ implies that $h_i(x_1) \in A^{\mathcal{U}_{q_i,\mathcal{O}}}$, and
2. for every role name r and variables $x_1, x_2 \in \mathsf{var}(q_{i+1})$, $(x_1, x_2) \in r^{\mathcal{U}_{q_{i+1},\mathcal{O}}}$ implies that $(h_i(x_1), h_i(x_2)) \in A^{\mathcal{U}_{q_i,\mathcal{O}}}$.

Since $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$, the function h_i^- cannot be a homomorphism from q_i to $\mathcal{U}_{q_{i+1},\mathcal{O}}$. Therefore, one of the following cases must apply:

1. there is a concept atom $A(x_1) \in q_i$ such that $h_i^-(x_1) \notin A^{\mathcal{U}_{q_{i+1},\mathcal{O}}}$;
2. there is a role atom $r(x_1, x_2) \in q_i$ such that $(h_i^-(x_1), h_i^-(x_2)) \notin r^{\mathcal{U}_{q_{i+1},\mathcal{O}}}$.

Thus, going from $\mathcal{U}_{q_i,\mathcal{O}}$ to $\mathcal{U}_{q_{i+1},\mathcal{O}}$, there must be a concept name A such that the number of variables in the interpretation of A strictly decreases, or a role name r such that the number of pairs of variables in the interpretation of r strictly decreases.

Let n_r and n_A be the numbers of role names and concept names in $\mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q_1)$, respectively. Since $q_1 \subseteq_{\mathcal{O}} q_j$, all concept and role names with non-empty interpretations in $\mathcal{U}_{q_j,\mathcal{O}}$ must be in $\mathsf{sig}(\mathcal{O}) \cup \mathsf{sig}(q_1)$. Therefore, the number of times variables occur in interpretations of concept names or role names in $\mathcal{U}_{q_j,\mathcal{O}}$ is bounded by

$$n_r \cdot |\mathsf{var}(q_j)|^2 + n_A \cdot |\mathsf{var}(q_j)|.$$

Since $|\mathsf{var}(q_j)| \leq |\mathsf{var}(q_T)|$ by Claim 1, the length of the sequence $q_j, \dots, q_k$ is thus bounded by

$$n_r \cdot |\mathsf{var}(q_j)|^2 + n_A \cdot |\mathsf{var}(q_j)| \leq (n_r + n_A) \cdot |\mathsf{var}(q_j)|^2 \leq (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|) \cdot |\mathsf{var}(q_T)|^2. \quad \square$$

Theorem 4.35 is restricted to *rooted* CQs. This is not an issue in this chapter, as all ELIQs are rooted, but it is still an interesting question if this restriction is necessary. The following example shows that the theorem fails already for Boolean CQs that do not use any role names under $\mathcal{ELI}$ ontologies.

Example 4.36. Let $n \geq 1$, let A_i, B_i be concept names for $1 \leq i \leq n$ and r a role name. Let $\mathcal{O}$ be an $\mathcal{ELI}$ ontology that contains the following concept inclusions, for all i with $1 \leq i \leq n$ and for all j with $1 \leq j < i$:

$$B_i \sqsubseteq \exists r.\top$$

$$\exists r^-.(A_1 \sqcap \cdots \sqcap A_{i-1} \sqcap B_i) \sqsubseteq A_i \qquad \exists r^-.(A_1 \sqcap \cdots \sqcap A_{i-1} \sqcap A_i) \sqsubseteq B_i$$

$$\exists r^-.B_i \sqcap B_j \sqsubseteq B_i \qquad \exists r^-.A_i \sqcap B_j \sqsubseteq A_i$$

Each subset of $\{A_i, B_i \mid 1 \leq i \leq n\}$ that contains exactly one of A_i, B_i for each i represents a binary number between 0 and 2^n-1, starting at $\{B_1, \dots, B_n\}$ for 0, $\{A_1, B_2, \dots, B_n\}$ for 1, and so on. Consider the ABox $\mathcal{A} = \{B_1(a), \dots, B_n(a)\}$. In $\mathcal{U}_{\mathcal{A},\mathcal{O}}$, there is an r-path starting at a of length $2^n - 1$, where the elements are labeled from 0 to $2^n - 1$. For any given set M_i that represents the number i, we construct the Boolean CQ

$$q_i() \leftarrow \bigwedge_{A \in M_i} A(x).$$

Then, the concept inclusions in $\mathcal{O}$ ensure that $q_j \subseteq_{\mathcal{O}} q_i$ if and only if $j \geq i$. Therefore, the sequence $q_0, q_1, \dots, q_{2^n-1}$ is a generalization sequence towards q_{2^n-1} under $\mathcal{O}$ of length 2^n. Furthermore, all q_i are $(q_{2^n-1}, \mathcal{O})$-minimal and $\|\mathcal{O}\| + \|q_{2^n-1}\|$ is polynomial in n.

Note that the ontology used in Example 4.36 is an $\mathcal{ELI}$ ontology. We will see in Chapter 5 that Theorem 4.35 can be generalized to queries that are not rooted, if we restrict the ontology to be formulated in $\mathcal{EL}^r$.

4.5 Obtaining an Initial Hypothesis

We now know that a learning algorithm can construct a generalization sequence towards the target query q_T under a *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontology by using the frontier construction and $\mathsf{minimize}_{\mathcal{O}}$, and that this is possible in polynomial time. What is missing, is a way to construct from the input signature Σ and ontology $\mathcal{O}$ an initial query q_1 of the generalization sequence The requirements we have for q_1 are that it must imply the target query q_T with $\mathsf{sig}(q_T) \subseteq \Sigma$ under the ontology $\mathcal{O}$, and that it is satisfiable under $\mathcal{O}$. Additionally, it should be of size polynomial in $\|q_T\|$, $\|\mathcal{O}\|$ and $|\Sigma|$ for the learning algorithm to run in polynomial time.

If we approach this directly and construct an ELIQ q_1 that guarantees that $q_1 \subseteq_{\mathcal{O}} q_T$ for all q_T with $\mathsf{sig}(q_T) \subseteq \Sigma$, we then require the *full tree* ELIQ of a certain depth, where every variable is labeled with all concept names, and every variable has a successor for every role in Σ. Unfortunately, such a full tree ELIQ must be of exponential size in $\|q_T\|$. Thus, we have to rely on membership queries to obtain a suitable ELIQ.

As a first step, we construct, given a *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontology $\mathcal{O}$ and signature Σ, directly a unary rooted CQ q^0_H that is satisfiable under $\mathcal{O}$ and implies any target query q_T with $\mathsf{sig}(q_T) \subseteq \Sigma$. Then, as a second step, we will see how we can use membership queries to generalize this q^0_H into an ELIQ q_1 such that q_1 still implies the target query.

Obtaining a Suitable Initial CQ

The construction of q_H^0 is simple if the ontology $\mathcal{O}$ contains no role disjointness constraints and no concept disjointness constraints. Simply set

$$q_H^0(x_0) \leftarrow \bigwedge_{A \in \Sigma \cap \mathsf{N_C}} A(x) \wedge \bigwedge_{r \in \Sigma \cap \mathsf{N_R}} r(x,x).$$

If, however, $\mathcal{O}$ contains role disjointness constraints, then the above construction yields a query that is not satisfiable under $\mathcal{O}$. In that case (but still without concept disjointness constraints in $\mathcal{O}$), we can still construct such a CQ, although it requires a bit more thought.

Let $\mathbf{R} = \{r_1, \dots, r_m\}$ be the set of all role names $r \in \Sigma \cap \mathsf{N_R}$ such that the concepts $\exists r.\top$ and $\exists r^-.\top$ are satisfiable under $\mathcal{O}$. If, for example, $\mathcal{O}$ contains $r \sqsubseteq s$ and $r \sqcap s \sqsubseteq \bot$, then $\exists r.\top$ is not satisfiable under $\mathcal{O}$ and the role name r is not included in $\mathbf{R}$.

To construct q_H^0, we use variables $x_0, \dots, x_{2m}$ and let K_{2m+1} be the *undirected* $2m+1$-*clique graph* that uses these variables as its vertices. It is known that for all odd $n \geq 1$, the n-clique K_n has at least $\frac{n-1}{2}$ Hamilton cycles that are pairwise edge-disjoint [ABS90]. We thus find in K_{2m+1} Hamilton cycles $P_1, \dots, P_m$ that are pairwise edge-disjoint. By directing the cycles, we may view each P_i as a set of directed edges (x_i, x_j). We then construct

$$q_H^0(x_0) \leftarrow \bigwedge_{\substack{A \in \Sigma \cap \mathsf{N_C} \\ 0 \leq i \leq 2m}} A(x_i) \wedge \bigwedge_{(x_i,x_j) \in P_1} r_1(x_i,x_j) \wedge \cdots \wedge \bigwedge_{(x_i,x_j) \in P_m} r_m(x_i,x_j).$$

By construction, q_H^0 has no multi-edges and thus satisfies all role disjointness constraints in $\mathcal{O}$. Moreover, every variable has exactly one r-successor and exactly one r-predecessor for every role name $r \in \mathbf{R}$ and hence satisfies all functionality assertions in $\mathcal{O}$. Additionally, q_H^0 implies every possible target ELIQ q_T, as q_T may only use role names from $\mathbf{R}$.

Example 4.37. For $\mathcal{O} = \{r \sqcap s \sqsubseteq \bot\}$ and $\Sigma = \{r, s\}$, the set $\mathbf{R}$ is $\{r, s\}$. The initial hypothesis that results from the construction is displayed in Figure 4.10, where the 5-clique is decomposed into two Hamilton cycles. Note how all role atoms are disjoint and functional.

If $\mathcal{O}$ contains at least one concept disjointness constraint $A \sqcap B \sqsubseteq \bot$, then the above construction of q_H^0 yields a query that is not satisfiable under $\mathcal{O}$. Indeed, then there is no single satisfiable CQ that implies every possible q_T. We can, however, obtain a suitable q_H^0 by using a single *equivalence query*. If $A \sqcap B \sqsubseteq \bot \in \mathcal{O}$, then the ELIQ $q(x_0) \leftarrow A(x_0) \wedge B(x_0)$ is not satisfiable under $\mathcal{O}$ and for every example

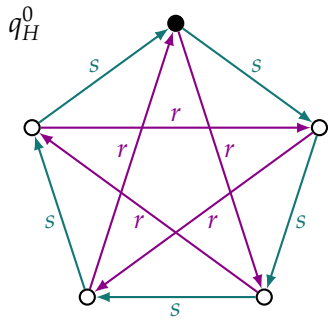

Figure 4.10: The query q_H^0 for $\mathbf{R} = \{r, s\}$. The s Hamilton cycle is marked in blue, the r Hamilton cycle is marked in purple.

$(\mathcal{A}, a)$, $\mathcal{A}, \mathcal{O} \not\models q(a)$. Hence, when the learning algorithm uses q in an equivalence query, the teacher is forced to return a counterexample $(\mathcal{A}, a)$ such that $\mathcal{A}, \mathcal{O} \models q_T(a)$. This counterexample can then be viewed as a CQ q_H^0, and it holds that $q_H^0 \subseteq_{\mathcal{O}} q_T$ by Lemma 3.7. Note that although the size of q_H^0 is then not bounded polynomially by $\|\mathcal{O}\|$, $\|q_T\|$ and $|\Sigma|$, this way to obtain q_H^0 also results in a polynomial time learning algorithm since the running time of learning algorithms may also depend polynomially on the size of the largest counterexample received.

We later show that this single equivalence query is necessary in this case, meaning that ELIQs are not polynomial time learnable under ontologies that contain concept disjointness constraints without using a single equivalence query.

Extracting an ELIQ

With a way to obtain a unary rooted CQ q_H^0 that implies q_T and is satisfiable under $\mathcal{O}$, it remains to show that we can extract from it an ELIQ with the same properties. For this, we need to make q_H^0 acyclic, that is, remove all cycles in q_H^0. A *cycle* in a CQ q is a sequence $R_1(x_1, x_2), \dots, R_n(x_n, x_1)$ of distinct role atoms in q such that $x_1, \dots x_n$ are distinct. Using the definition of acyclicity, it is easy to verify that a CQ is acyclic (defined through the underlying graph of $\mathcal{A}_q$) if and only if it contains no cycles.

For removing all cycles from q_H^0 while maintaining $q_H^0 \subseteq_{\mathcal{O}} q_T$, we define a new subroutine called $\mathsf{extract}_{\mathrm{ELIQ}}$ that we use as part of our learning algorithm. It takes as input the ontology $\mathcal{O}$ and a unary CQ $q(x_0)$ that is satisfiable under $\mathcal{O}$, and satisfies $q \subseteq_{\mathcal{O}} q_T$. It then computes an ELIQ q', such that $q \subseteq_{\mathcal{O}} q'$ and $q' \subseteq_{\mathcal{O}} q_T$ by repeatedly doubling the length of cycles in q and then using $\mathsf{minimize}_{\mathcal{O}}$ to attain $(q_T, \mathcal{O})$-minimality. A procedure similar to $\mathsf{extract}_{\mathrm{ELIQ}}$ is used in [tCD22] to obtain acyclic queries in the case without ontologies. Here, role inclusions need to be taken into account.

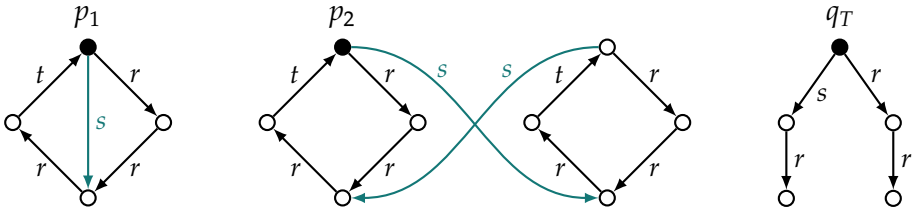

Figure 4.11: The CQ p_1 contains the cycle $r(x_0, x_1), s(x_1, x_2), t(x_2, x_0)$ which is removed by *Double cycle* (p_2), since the cycle is not necessary for q_T.

The subroutine $\mathsf{extract}_{\mathrm{ELIQ}}$ starts by setting $p = \mathsf{minimize}_{\mathcal{O}}(q)$ and then returns the result of exhaustively applying the following operations:

Double cycle. Choose a role atom $r(x, y) \in p$ that is part of a cycle and such that there is no $s(x, y) \in p$ with $\mathcal{O} \models s \sqsubseteq r$ and $r \neq s$. Then, add a disjoint copy p' of p to p and let x', y' be the copies of x, y in p'. Remove the atoms $r(x, y), r(x', y')$ and add the atoms $r(x, y'), r(x', y)$. Apply $\mathsf{minimize}_{\mathcal{O}}$ to the result.

Drop double edge. Choose a role atom $r(x, y) \in p$ such that there is a role atom $s(x, y) \in p$ with $\mathcal{O} \models r \equiv s$ and $s \neq r$ and remove $r(x, y)$.

Example 4.38. Consider the CQs p_1, p_2, q_T displayed in Figure 4.11 and $\mathcal{O} = \emptyset$. It holds that $p_1 \subseteq_{\mathcal{O}} q_T$, but p_1 is not acyclic. Applying *Double cycle* to p_1 first results in the CQ p_2 with $p_2 \subseteq_{\mathcal{O}} q_T$. Then, $\mathsf{minimize}_{\mathcal{O}}$ is applied to p_2, which in this case, removes the remaining cycles.

It remains to show that $\mathsf{extract}_{\mathrm{ELIQ}}$ always results in an ELIQ q with $q \subseteq_{\mathcal{O}} q_T$, and that it only applies a polynomial number of operations before terminating. For this, we will view the intermediate steps between applications of *Double cycle* as a sequence of queries $p_0, p_1, \ldots$ and show that they form a generalization sequence towards q_T under $\mathcal{O}$. Since all p_i are the result of applying $\mathsf{minimize}_{\mathcal{O}}$, they are all $(q_T, \mathcal{O})$-minimal, and we can apply Theorem 4.35 to show that $\mathsf{extract}_{\mathrm{ELIQ}}$ terminates after a polynomial number of steps. We show that $p_0, p_1, \ldots$ is a generalization sequence by relating p_i to p_{i+1} using $\mathcal{ELI}$ simulations.

Definition 4.39 ($\mathcal{ELI}$ simulation)**.** An $\mathcal{ELI}$ *simulation* from interpretation $\mathcal{I}_1$ to interpretation $\mathcal{I}_2$ is a relation $S \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ such that for all $(d_1, d_2) \in S$:

1. for all $A \in \mathsf{N_C}$: if $d_1 \in A^{\mathcal{I}_1}$, then $d_2 \in A^{\mathcal{I}_2}$;

2. for all $r \in \mathsf{N_R}$ and $R \in \{r, r^-\}$: if there is some $d_1' \in \Delta^{\mathcal{I}_1}$ with $(d_1, d_1') \in R^{\mathcal{I}_1}$, then there is $d_2' \in \Delta^{\mathcal{I}_2}$ such that $(d_1', d_2') \in S$ and $(d_2, d_2') \in R^{\mathcal{I}_2}$.

If there is an $\mathcal{ELI}$ simulation S from an interpretation $\mathcal{I}_1$ to an interpretation $\mathcal{I}_2$ with $(d_1, d_2) \in S$, we write $\mathcal{I}_1, d_1 \preceq_{\mathcal{ELI}} \mathcal{I}_2, d_2$. As usual, as we can view ABoxes as finite interpretations, we also define this notation for ABoxes. The important property that connects $\mathcal{ELI}$ simulations to ELIQs is given in the following lemma, the proof is standard and omitted.

Lemma 4.40. *Let $\mathcal{O}$ be an $\mathcal{ELIHF}_\bot$ ontology, $\mathcal{A}_1$, $\mathcal{A}_2$ ABoxes such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are satisfiable under $\mathcal{O}$. If $\mathcal{A}_1, a_1 \preceq_{\mathcal{ELI}} \mathcal{A}_2, a_2$, then for all ELIQs q, $\mathcal{A}_1, \mathcal{O} \models q(a_1)$ implies $\mathcal{A}_2, \mathcal{O} \models q(a_2)$.*

Note that Lemma 4.40 does not hold for all CQs in place of just ELIQs. Consider the ABoxes $\mathcal{A}_1 = \{r(a, a)\}$ and $\mathcal{A}_2 = \{r(b_1, b_2), r(b_2, b_1)\}$, the CQ $q(x) \leftarrow r(x, x)$ and $\mathcal{O} = \emptyset$. Then $\mathcal{A}_1, \mathcal{O} \models q(a)$ and $\mathcal{A}_2, \mathcal{O} \not\models q(b_1)$, but $\mathcal{A}_1, a \preceq_{\mathcal{ELI}} \mathcal{A}_2, b_1$ as witnessed by the $\mathcal{ELI}$ simulation $S = \{(a, b_1), (a, b_2)\}$.

Lemma 4.41. *Let $\mathcal{O}$ be a DL-Lite$^{\mathcal{HF}^-}_{\text{core}}$ or DL-Lite$^{\mathcal{F}}_{\text{horn}}$ ontology, q_T an ELIQ and q a unary CQ with $q \subseteq_{\mathcal{O}} q_T$ that is satisfiable under $\mathcal{O}$. Then, $\mathsf{extract}_{\text{ELIQ}}(\mathcal{O}, q)$ runs in time polynomial in $\|\mathcal{O}\| + \|q\| + \|q_T\|$ and returns an ELIQ q' that is $(q_T, \mathcal{O})$-minimal and satisfies $q \subseteq_{\mathcal{O}} q' \subseteq_{\mathcal{O}} q_T$.*

Proof. We first proof that the sequence $p_0, p_1, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$, and then apply Theorem 4.35 to show that this sequence must terminate after a polynomial number of steps. Additionally, if this sequence terminates, then both *Double cycle* and *Drop double edge* are no longer applicable, and the result must be an ELIQ.

First, note that *Double cycle* preserves satisfiability under $\mathcal{O}$. Since the input q to $\mathsf{extract}_{\text{ELIQ}}$ is assumed to be satisfiable under $\mathcal{O}$, all p_i are satisfiable under $\mathcal{O}$ as well, and we can use the characterization of query containment in terms of homomorphisms to the universal model provided in Lemma 3.7. We do this without further notice below.

We start by showing $p_i \subseteq_{\mathcal{O}} p_{i+1}$ for all $i \geq 0$. Let p_i' be the result of applying *Double cycle* to p_i before using $\mathsf{minimize}_{\mathcal{O}}$. Then $p_{i+1} = \mathsf{minimize}_{\mathcal{O}}(p_i')$. It suffices to show $p_i \subseteq_{\mathcal{O}} p_i'$. To achieve this, in turn, it is enough to point out that we obtain a homomorphism h_i from p_i' to p_i with $h_i(x_0) = x_0$ by setting $h_i(x) = x$ for all $x \in \mathsf{var}(p_i)$ and $h_i(x') = x$ for all variables x' in the disjoint copy of p_i that is added in *Double Cycle*. We shall reuse h_i below and call it the *natural homomorphism* from p_i' to p_i.

Next, we show that $p_i \subseteq_{\mathcal{O}} q_T$ for all $i \geq 0$ by induction on i. In the induction start, $p_0 = \mathsf{minimize}_{\mathcal{O}}(p)$, where p is the input to $\mathsf{extract}_{\text{ELIQ}}$. Since $p \subseteq_{\mathcal{O}} q_T$, applying Lemma 4.32 yields $p_0 \subseteq_{\mathcal{O}} q_T$. Now assume that $p_i \subseteq_{\mathcal{O}} q_T$ and thus $\mathcal{A}_{p_i}, \mathcal{O} \models q_T(x_0)$. Let again p_i' be the result of applying *Double cycle* to p_i before using $\mathsf{minimize}_{\mathcal{O}}$. Again it suffices to show $p_i' \subseteq_{\mathcal{O}} q_T$. Define the relation

$$S = \{(h_i(x), x) \mid x \in \mathsf{var}(p_i')\}$$

where h_i is the natural homomorphism from p_i' to p_i. By construction of p_i', S is an $\mathcal{ELI}$ simulation from $\mathcal{A}_{p_i}$ to $\mathcal{A}_{p_i'}$ with $(x_0, x_0) \in S$. Thus, $\mathcal{A}_{p_i'}, \mathcal{O} \models q_T(x_0)$ by Lemma 4.40, and $p_i' \subseteq_{\mathcal{O}} q_T$ follows as required.

It remains to show that $p_{i+1} \not\subseteq_{\mathcal{O}} p_i$ for all $i \geq 0$. Similarly to what was done above, it suffices to show that $p_i' \not\subseteq_{\mathcal{O}} p_i$ where p_i' is the result of applying *Double cycle* to p_i. Assume to the contrary that $p_i' \subseteq_{\mathcal{O}} p_i$ for some i. Then, there is a homomorphism g from p_i to $\mathcal{U}_{p_i', \mathcal{O}}$ with $g(x_0) = x_0$. Composing g with the extension h_i^+ of the natural homomorphism h_i to a homomorphism from $\mathcal{U}_{p_i', \mathcal{O}}$ to $\mathcal{U}_{p_i, \mathcal{O}}$, which exists by Lemma 3.8 yields a homomorphism $\widehat{g}$ from p_i to $\mathcal{U}_{p_i, \mathcal{O}}$ with $\widehat{g}(x_0) = x_0$.

Let $R_1(y_1, y_2), \dots, R_n(y_n, y_1)$ be the cycle that was expanded in the construction of p_i' and consider the set Γ of all sets of variables that form a cycle of length n in $\mathcal{U}_{p_i, \mathcal{O}}$. For example, $\{y_1, \dots, y_n\} \in \Gamma$.

Let $\{x_1, \dots, x_n\}$ be any element of Γ. We show that $\{\widehat{g}(x_1), \dots, \widehat{g}(x_n)\} \in \Gamma$. If for some x_i, $\widehat{g}(x_i)$ is a proper trace, then Lemma 4.21 implies that p_i is not $(q_T, \mathcal{O})$-minimal, a contradiction. Since $\widehat{g}$ is a homomorphism, it thus suffices to show that $\widehat{g}(x_1), \dots, \widehat{g}(x_n)$ are all pairwise different. Assume the contrary. Then there are x_j and x_k with $x_j \neq x_k$ and $\widehat{g}(x_j) = \widehat{g}(x_k)$, implying that $\widehat{g}$ is not injective. This, in turn, implies that there is an $x \in \mathsf{var}(p_i)$ with $x \notin \mathsf{img}(\widehat{g})$. It then follows from Lemma 4.21 that p_i is not $(q_T, \mathcal{O})$-minimal, a contradiction.

Therefore, we can define a function $f \colon \Gamma \to \Gamma$ by setting for all $\{x_1, \dots, x_n\} \in \Gamma$

$$f(\{x_1, \dots, x_n\}) = \{\widehat{g}(x_1), \dots, \widehat{g}(x_n)\}.$$

Assume that there are sets $\gamma, \gamma' \in \Gamma$ with $\gamma \neq \gamma'$ and $f(\gamma) = f(\gamma')$. Since $\gamma \neq \gamma'$ and $|\gamma| = |\gamma'|$, there must be a variable $x \in \gamma$ with $x \notin \gamma'$. Since $f(\gamma) = f(\gamma')$, there is a variable $x' \in \gamma'$ with $\widehat{g}(x) = \widehat{g}(x')$, and clearly $x' \neq x$. This again contradicts $(q_T, \mathcal{O})$-minimality of p_i via Lemma 4.21. Thus, f is a bijection from Γ to Γ.

Since Γ is finite, it follows that there must be a $j \geq 1$ such that $f^j(\{y_1, \dots, y_n\}) = \{y_1, \dots, y_n\}$. By definition of f this implies that $\{\widehat{g}^j(y_1), \dots, \widehat{g}^j(y_n)\} = \{y_1, \dots, y_n\}$. Recall that $\widehat{g}$ is the composition of the homomorphism g from p_i to $\mathcal{U}_{p_i', \mathcal{O}}$ and the homomorphism h_i^+ from $\mathcal{U}_{p_i', \mathcal{O}}$ to $\mathcal{U}_{p_i, \mathcal{O}}$. Since $(q_T, \mathcal{O})$-minimality of p_i implies that $\widehat{g}$ is injective by Lemma 4.21, g must also be injective. Thus, composing $\widehat{g}^{j-1}$ and g yields an injective homomorphism g' that maps the cycle $\{y_1, \dots, y_n\}$ in p_i to some subset of the expanded cycle $\{y_1, y_1', \dots, y_n, y_n'\}$ in $\mathcal{U}_{p_i', \mathcal{O}}$. We distinguish cases.

First, consider the case where $\{g'(y_1), \dots, g'(y_n)\} = \{y_1, \dots, y_n\}$. By the construction of p_i' from p_i, the restriction of $\mathcal{U}_{p_i', \mathcal{O}}$ to $\{y_1, \dots, y_n\}$ contains one less role than the restriction of $\mathcal{U}_{p_i, \mathcal{O}}$ to $\{y_1, \dots, y_n\}$, implying that g' cannot be an injective homomorphism, leading to a contradiction. The case where $\{g'(y_1), \dots, g'(y_n)\} = \{y_1', \dots, y_n'\}$ is analogous.

The remaining case is that $\{g'(y_1), \dots, g'(y_n)\}$ contains both variables of the form y_j and y_j'. Then, there must be two different atoms in the cycle $R_1(y_1, y_2), \dots, R_n(y_n, y_1)$

Algorithm 4.1: Learning algorithm for ELIQs under *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontologies

Input A signature Σ and a *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontology $\mathcal{O}$ in normal form
Output An ELIQ q_H such that $q_H \equiv_{\mathcal{O}} q_T$

$q_H^0 :=$ initial-CQ$(\Sigma, \mathcal{O})$
$q_H :=$ extract$_{\text{ELIQ}}(\mathcal{O}, q_H^0)$
while there is a $q_F \in F_{q_H}$ with $q_F \subsetneq_{\mathcal{O}} q_T$ **do**
    $q_H :=$ minimize$_{\mathcal{O}}(q_F)$
end while
return q_H

that are mapped by g' to the role atoms $r(x, y'), r(x', y)$ that were added by *Double cycle* to connect the disjoint copy of p_i. However, since $h_i(x') = h_i(x)$ and $h_i(y') = h_i(y)$, this implies that the composition of g' and h_i^+ is a non-injective homomorphism from p_i to $\mathcal{U}_{p_i,\mathcal{O}}$, again contradicting $(q_T, \mathcal{O})$-minimality of p_i. □

The purpose of *Drop double edge* is to deal with cycles that cannot be handled by *Double cycle* due to the "such that there is no $s(x, y) \in p$ with $\mathcal{O} \models s \sqsubseteq r$ and $r \neq s$" condition, which, in turn, is necessary for *Drop double edge* to produce a generalization sequence. It follows directly from the definition of the operation that it can be applied at most $\|p\|$ times.

Note that the number of applications of *Double cycle* does not depend on the ontology language. However, the running time of extract$_{\text{ELIQ}}$ depends on the ontology language, as it applies minimize$_{\mathcal{O}}$. Per Lemma 4.32, minimize$_{\mathcal{O}}$ runs in polynomial time if $\mathcal{O}$ is an *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ or *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ ontology, but not if $\mathcal{O}$ is an $\mathcal{ELI}$ or $\mathcal{ELIHF}_\bot$ ontology.

Therefore, we can use extract$_{\text{ELIQ}}(\mathcal{O}, q_H^0)$ in a learning algorithm to produce an initial hypothesis ELIQ to start the generalization sequence that approaches q_T.

4.6 The Learning Algorithm for ELIQs

In Sections 4.3 to 4.5 we obtained the necessary pieces for a learning algorithm of ELIQs under *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontologies. The resulting algorithm is Algorithm 4.1. It takes as input a signature Σ and an ontology in normal form, produces an initial CQ q_H^0 according to Section 4.5, and then uses extract$_{\text{ELIQ}}$, minimize$_{\mathcal{O}}$ and the frontier construction to produce an ELIQ that is equivalent to q_T under $\mathcal{O}$.

In order to show that Algorithm 4.1 is indeed a polynomial time learning algorithm, we use the facts that extract$_{\text{ELIQ}}$ and minimize$_{\mathcal{O}}$ run in polynomial time

(Lemma 4.32 and Lemma 4.41), and show that the assignments to q_H form a generalization sequence towards q_T under $\mathcal{O}$, which allows us then to apply Theorem 4.35.

Theorem 4.42. *ELIQs are polynomial time learnable under DL-Lite$_{\text{core}}^{\mathcal{HF}^-}$ ontologies using only membership queries. If the ontology contains concept disjointness constraints, then this requires one additional equivalence query.*

Proof. We show that Algorithm 4.1 is a polynomial time learning algorithm for ELIQs under DL-Lite$_{\text{core}}^{\mathcal{HF}^-}$ ontologies in normal form.

Let Σ be a signature and $\mathcal{O}$ a DL-Lite$_{\text{core}}^{\mathcal{HF}^-}$ ontology in normal form. Then an initial CQ q_H^0 satisfiable under $\mathcal{O}$ and such that $q_H^0 \subseteq_{\mathcal{O}} q_T$ can be obtained from Σ and $\mathcal{O}$ in polynomial time, as described in Section 4.5. Further, let $q_1, q_2, \ldots$ be the sequence of queries that is assigned to q_H during a run of Algorithm 4.1. We aim to show that $q_1, q_2, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$.

First, we show that $q_i \subseteq_{\mathcal{O}} q_T$ for all $i \geq 1$ by induction on i. Since $q_H^0 \subseteq_{\mathcal{O}} q_T$, Lemma 4.41 implies that $q_1 \subseteq_{\mathcal{O}} q_T$. Now, assume that $q_i \subseteq_{\mathcal{O}} q_T$ and that there is a query q_{i+1} in the sequence. Then there is a $q_i' \in F_{q_i}$ such that $q_i' \subseteq_{\mathcal{O}} q_T$ and $q_{i+1} = \mathsf{minimize}_{\mathcal{O}}(q_i')$. Lemma 4.32 then implies $q_{i+1} \subseteq_{\mathcal{O}} q_T$.

Then, we show that $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$ for all $i \geq 1$. Consider any i. Again, there is a $q_i' \in F_{q_i}$ with $q_{i+1} = \mathsf{minimize}_{\mathcal{O}}(q_i')$. By Definition 4.13, $q_i \subseteq_{\mathcal{O}} q_i'$ and $q_i' \not\subseteq_{\mathcal{O}} q_i$. Lemma 4.32 then implies that $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$.

Hence, $q_1, q_2, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$. Since all q_i are $(q_T, \mathcal{O})$-minimal, Theorem 4.35 implies that the sequence has length at most $|\mathsf{var}(q_T)|^3 \cdot (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$. Thus, the sequence has a last element q_n, that is returned by Algorithm 4.1. The while loop condition implies that there is no $q_n' \in F_{q_n}$ with $q_n' \subseteq_{\mathcal{O}} q_T$. Therefore, $q_n \equiv_{\mathcal{O}} q_T$ by Definition 4.13.

As $\mathsf{minimize}_{\mathcal{O}}$ runs in polynomial time by Lemma 4.32, $\mathsf{extract}_{\text{ELIQ}}$ runs in polynomial time by Lemma 4.41, and F_{q_H} can be computed in polynomial time by Theorem 4.23, and the number of loop iterations is bounded by a polynomial, Algorithm 4.1 runs in polynomial time in $\|\mathcal{O}\|$, $|\Sigma|$ and $\|q_T\|$. □

As mentioned in Section 4.5, the single equivalence query in the case of disjointness constraints is really necessary. The following theorem shows that we cannot learn the simple class of conjunctions of atomic queries using only a polynomial number of membership queries under disjointness constraints. A *disjointness ontology* is an ontology that contains only concept disjointness constraints.

Theorem 4.43. *Conjunctions of atomic queries are not polynomial query learnable under disjointness ontologies using only membership queries.*

Proof. We follow the same strategy as the proof of Theorem 4.5. For every $n \geq 1$, let

$$\mathcal{O}_n = \{A_i \sqcap B_i \sqsubseteq \bot \mid 1 \leq i \leq n\}$$

and

$$S_n = \{q(x) \leftarrow \alpha_1(x) \wedge \ldots \wedge \alpha_n(x) \mid \alpha_i \in \{A_i, B_i\} \text{ for all } i \text{ with } 1 \leq i \leq n\}.$$

Note that S_n is a frontier of $\bot$ under $\mathcal{O}_n$, if only conjunctions of atomic queries using the concept names A_i and B_i for all $1 \leq i \leq n$ are considered for Condition 2[5]. Clearly, S_n contains 2^n queries.

Assume to the contrary of what is to be shown that conjunctions of atomic queries are polynomial query learnable under disjointness ontologies using only membership queries. Then there exists a learning algorithm and polynomial p such that the number of membership queries needed to identify a target query q_T is bounded by $p(n_\Sigma, n_\mathcal{O}, n_{q_T})$, where n_Σ is the size of the signature Σ, $n_\mathcal{O}$ is the size of the ontology and n_{q_T} is the size of the target query. We choose n such that $2^n > p(2n, \|\mathcal{O}_n\|, r(n))$, where r is a polynomial such that every query $q \in S_m$ satisfies $\|q\| = r(m)$.

Now, consider a membership query posed by the learning algorithm with the data example $(\mathcal{A}, a)$. The teacher responds as follows:

1. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for no $q \in S_n$, then answer *no*;
2. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for a single $q \in S_n$, then answer *no* and remove q from S_n;
3. if $\mathcal{A}, \mathcal{O}_n \models q(a)$ for more than one $q \in S_n$, then answer *yes*.

Note that the third response is consistent since $\mathcal{A}$ must then contain $A_i(a)$ and $B_i(a)$ for some i and thus $\mathcal{A}$ is not satisfiable under $\mathcal{O}_n$. Moreover, the answers are always correct with respect to the updated set S_n. Thus, the learner cannot distinguish the remaining candidate queries by answers to queries posed so far.

It follows that the learning algorithm removes at most $p(2n, \|\mathcal{O}_n\|, r(n))$ queries from S_n. By the choice of n, at least two candidate concepts remain in S_n after the algorithm is finished. Thus, the learner cannot distinguish between them, and we have derived a contradiction. □

4.7 Discussion

In this chapter, we investigated the learnability of ELIQs under ontologies using only membership queries. The results can be summarized as follows. ELIQs are

- *not learnable* under ontology languages that contain $\mathcal{F}_{\text{core}}$ (Theorem 4.6);

[5] In fact, it can be shown similarly as in the proof of Theorem 4.28 that S_n is contained in any frontier of $\bot$ under $\mathcal{O}_n$. Hence, $\bot$ does not have polynomially sized frontiers under disjointness ontologies.

- *not polynomial query learnable* under ontology languages that can express conjunctions such as *DL-Lite*$_{\text{horn}}$ or $\mathcal{EL}$ (Theorem 4.5);
- *polynomial time learnable* under *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies, using at most one additional equivalence query (Theorem 4.42).

Recall that these results also apply to learning $\mathcal{ELI}$ concepts using membership queries.

The learning algorithm for ELIQs under *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies is based on *frontiers* of ELIQs. The existence of frontiers of ELIQs under *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies that can be computed in polynomial time is an interesting result on its own, and may have applications in other query engineering tasks. For showing that the learning algorithm runs in polynomial time, we used the notion of *generalization sequences*, and proved that generalization sequences of $(q_T, \mathcal{O})$-minimal rooted queries are of at most polynomial length, even under $\mathcal{ELIHF}_\bot$ ontologies. This too might be interesting for other query engineering tasks.

Next, we discuss some properties of Algorithm 4.1 and point to possible future directions.

What happens when q_T is not an ELIQ? One of the basic assumptions of the learning algorithm is that q_T is an ELIQ or rather that the membership queries are answered according to some ELIQ. This cannot always be guaranteed in practical scenarios. If we consider the scenario where q_T is not an ELIQ but a rooted CQ, then there are two different behaviors of Algorithm 4.1. Recall that rooted CQs are not polynomial query learnable. If there is no ELIQ q such that $q \subseteq_{\mathcal{O}} q_T$ because q_T contains directed cycles, then Algorithm 4.1 gets stuck in $\text{extract}_{\text{ELIQ}}$. This can easily be detected, and the algorithm can abort. If there is such an ELIQ, then Algorithm 4.1 produces the most general ELIQ q such that $q \subseteq_{\mathcal{O}} q_T$. This result can be useful, as it provides information about q_T. However, in the scenario where membership queries are not answered consistently with some CQ, it is not guaranteed that Algorithm 4.1 produces any useful result. More research is necessary to develop learning algorithms that can cope with inconsistently answered membership queries.

The $\mathcal{EL}$ subsumption lattice. As we can view ELQs as $\mathcal{EL}$ concepts, we can view Algorithm 4.1 applied to an ELQ target query as a way to traverse the $\mathcal{EL}$ subsumption lattice by using *upwards neighbors* (under the empty ontology). Kriegel observed that concepts of size n have up to n-fold exponential distances in this lattice and identified this as an obstacle to $\mathcal{EL}$ learning algorithms [Kri21]. Algorithm 4.1 avoids this obstacle by minimizing the current hypothesis using membership queries. Indeed, Theorem 4.35 fails if the $(q_T, \mathcal{O})$-minimality requirement is

dropped. Intuitively, we can view Algorithm 4.1 as working on the structure of $\mathcal{EL}$ concepts of size at most $\|q_T\|$.

Concept membership queries. In some scenarios, it might make sense to restrict membership queries further. For example, one might be interested in learning $\mathcal{ELI}$ concepts using membership queries that use concept examples and not data examples. In this setting, the target is an $\mathcal{ELI}$ concept C_T and a membership query is asked with an $\mathcal{ELI}$ concept C to which the teacher responds *yes* if $\mathcal{O} \models C \sqsubseteq C_T$ and *no* otherwise. The basic principle of Algorithm 4.1 still works in this case, as the frontier construction and $\mathsf{minimize}_{\mathcal{O}}$ produce only ELIQs that we can view as $\mathcal{ELI}$ concepts. However, $\mathsf{extract}_{\mathrm{ELIQ}}$ needs to ask membership queries with cyclic data examples to obtain an initial hypothesis, and we conjecture that this cannot be avoided. Consider $\mathcal{O} = \emptyset$ and the set

$$S_n = \{\exists r_1. \cdots \exists r_n.\top \mid r_i \in \{r, s\} \text{ for } 1 \leq i \leq n\}.$$

A similar proof to the proof of Theorem 4.5 could show that an adversarial teacher can answer every membership query with concept C with *no*, and only needs to remove $\|C\|$ concepts from S_n. Since there are 2^n concepts in S_n, a learning algorithm cannot identify every concept in S_n using membership queries of polynomial size.

Fixed ontologies. The lower bounds in Theorem 4.5 and Theorem 4.43 rely on ontologies and signatures that are chosen based on the learning algorithm. Since in practice ontologies are relatively small and seldom change, it may make sense to consider a modified version of polynomial time learning in which the ontology is fixed and the running time of a learning algorithm need not be polynomial in the size of the ontology. As Theorem 4.5 and Theorem 4.43 do not apply in this modified setting, determining the learnability of ELIQs in that setting is a possible direction to extend the work of this chapter.

Known size of q_T. The lower bound in Theorem 4.6 uses a fixed ontology, but the structure of the used queries is simple. If the size of the target query were known, a learning algorithm could quickly determine it. A possible future direction is to consider learnability of ELIQs under $\textit{DL-Lite}^{\mathcal{F}}_{\mathrm{core}}$ ontologies, where the size of the target query is known to the learning algorithm. We conjecture that the proof of Theorem 4.6 can be modified for this setting by using a binary-tree-like structure to show that polynomial query learnability remains impossible in this case.

Conjunction-free $\mathcal{EL}$ ontologies. A different possible extension is to consider restrictions of the DL $\mathcal{EL}$ that cannot use conjunctions, as this avoids the lower bound

of Theorem 4.28. It is an interesting question whether ELIQs have frontiers under these ontologies, and whether this enables polynomial time learning. However, it is questionable if such an $\mathcal{EL}$ fragment is found in practice. Note that it does not make sense to restrict $\mathcal{ELI}$ to be conjunction-free, as a conjunction $A_1 \sqcap A_2 \sqsubseteq B$ can be expressed as

$$A_1 \sqsubseteq \exists r_{A_1}.\top,\ \exists r^-_{A_1}.A_2 \sqsubseteq A_2,\ \exists r^-_{A_1}.A_2 \sqsubseteq B,$$

without the use of explicit conjunction.

c-acyclic queries. Ten Cate and Dalmau show that frontiers of a larger class of CQs, namely c-acyclic CQs, can be constructed in polynomial time under the empty ontology. The frontiers themselves consist of CQs that are not c-acyclic, but can be used as part of a learning algorithm by applying a procedure like $\mathsf{extract}_{\mathrm{ELIQ}}$. We conjecture that this result can be extended to the case with *DL-Lite*$^{\mathcal{HF}^-}_{\mathrm{core}}$ ontologies, generalizing Theorem 4.23. If frontiers of c-acyclic CQs under *DL-Lite*$^{\mathcal{HF}^-}_{\mathrm{core}}$ ontologies can be computed in polynomial time, this could be the basis for polynomial time learnability of c-acyclic CQs under *DL-Lite*$^{\mathcal{HF}^-}_{\mathrm{core}}$ ontologies using only membership queries, which also includes queries with multiple answer variables.

In the next chapter, we consider learning algorithms that use both membership queries and equivalence queries.

Chapter 5

Learning with Membership and Equivalence Queries

In Chapter 4, we have observed that learning with only membership queries has its limits. Under many extensions of *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ that contain $\mathcal{EL}$ or *DL-Lite*$_{\text{horn}}$, every correct learning algorithm needs to ask at least an exponential number of membership queries to identify a target query in the worst case. In this chapter, we consider more powerful learning algorithms that are in addition to membership queries able to ask *equivalence queries*. We show that equivalence queries allow polynomial time learning of queries under ontologies in cases where membership queries alone do not suffice.

Recall that a learning algorithm for a query class $\mathcal{Q}$ that attempts to identify a target query $q_T \in \mathcal{Q}$ under an ontology $\mathcal{O}$ asks an equivalence query by handing a hypothesis query $q_H \in \mathcal{Q}$ to the teacher. The teacher responds with *yes* if q_H is equivalent to q_T under $\mathcal{O}$ and otherwise returns a counterexample, that is a data example $(\mathcal{A}, \overline{a})$ such that $\mathcal{A}, \mathcal{O} \models q_H(\overline{a})$ and $\mathcal{A}, \mathcal{O} \not\models q_T(\overline{a})$, or the other way around. As soon as the learner receives a *yes* as an answer to an equivalence query, it can terminate as it has identified the target query.

Example 5.1. Consider the ontology $\mathcal{O} = \{A \sqsubseteq B\}$ and the target query $q_T(x_0) \leftarrow r(x_0, x_1) \wedge B(x_1)$. If the learning algorithm asks an equivalence query with the hypothesis $q_H(x_0) \leftarrow B(x_0)$, then the teacher could respond with the counterexample $(\mathcal{A}_1, a)$ with $\mathcal{A}_1 = \{A(a)\}$ since $\mathcal{A}_1, \mathcal{O} \models q_H(a)$ and $\mathcal{A}_1, \mathcal{O} \not\models q_T(a)$, or with the counterexample $(\mathcal{A}_2, a)$ with $\mathcal{A}_2 = \{r(a, b), r(b, b), A(b)\}$ since $\mathcal{A}_2, \mathcal{O} \models q_T(a)$ and $\mathcal{A}_2, \mathcal{O} \not\models q_H(a)$.

Equivalence queries allow learning algorithms to circumvent many of the obstacles identified in Chapter 4.

Example 5.2. Consider, for some $n \geq 1$, the set

$$S_n = \{q(x) \leftarrow \alpha_1(x) \wedge \ldots \wedge \alpha_n(x) \mid \alpha_i \in \{A_i, B_i\} \text{ for all } i \text{ with } 1 \leq i \leq n\},$$

and the ontology

$$\mathcal{O}_n = \{A_i \sqcap B_i \sqsubseteq A_1 \sqcap B_1 \sqcap \cdots \sqcap A_n \sqcap B_n' \mid 1 \leq i \leq n\}.$$

from the proof of Theorem 4.5. We have shown that every learning algorithm requires in the worst case $2^n - 1$ membership queries to identify a target query from the set S_n. However, a single equivalence query with the hypothesis $q_H(x_0) \leftarrow A_1(x_0) \wedge B_1(x_0)$, forces the teacher to return a counterexample $(\mathcal{A}, a)$ such that $\mathcal{A}, \mathcal{O} \models q_T(a)$ and $\mathcal{A}, \mathcal{O} \not\models q(a)$ for all $q \in S_n$ with $q \neq q_T$. Hence, a target query $q_T \in S_n$ can be identified with a single equivalence query.

In this chapter, we consider polynomial learnability of queries under $\mathcal{EL}^r$, $\mathcal{ELI}$, and $\textit{DL-Lite}_{\text{horn}}^{\mathcal{F}-}$ ontologies, all of which extend $\textit{DL-Lite}_{\text{core}}$ with conjunctions and hence make polynomial time learning of queries with only membership queries impossible.

To show learnability with both membership and equivalence queries, we use learning algorithms that follow a similar structure as the ones in Chapter 4. Beginning at an initial hypothesis, they produce a sequence of queries that are more specific than the target query q_T and approach q_T step-by-step. The main difference lies in how this sequence is produced, as frontiers of polynomial size are not available under the ontology languages we consider in this chapter. Instead, the learning algorithms use the counterexamples provided by equivalence queries to update hypotheses.

Structure of This Chapter

We begin in Section 5.1 by discussing how counterexamples can be used in learning algorithms. While it is relatively easy to use counterexamples to update a hypothesis in learning algorithms under the empty ontology, ontologies complicate the situation. This is because *least general generalizations* do not always exist under ontologies.

Then, in Section 5.2 we turn to learning under $\textit{DL-Lite}_{\text{horn}}^{\mathcal{F}-}$ ontologies that in contrast to $\textit{DL-Lite}_{\text{core}}^{\mathcal{F}-}$ ontologies allow the use of conjunctions in concept inclusions. We show that ELIQs are polynomial time learnable under $\textit{DL-Lite}_{\text{horn}}^{\mathcal{F}-}$ ontologies using both membership queries and equivalence queries. For this, we introduce *guided generalizations* and combine them with results from Chapter 4 concerning minimization and generalization sequences.

In Section 5.3, we consider learning under $\mathcal{EL}^r$, which extends $\textit{DL-Lite}_{\text{core}}$ with conjunction and qualified existential restrictions, but restricts the use of inverse roles. We show that ELQs are polynomial time learnable under $\mathcal{EL}^r$ ontologies using membership queries and equivalence queries, and extend this result to so-called *symmetry-free* ELIQs and *chordal symmetry-free* CQs of fixed arity. For this, we need to generalize our results regarding minimization and generalization sequences to CQs that are not rooted and have multiple answer variables, which brings some technical challenges.

Algorithm 5.1: Learning algorithm for CQs under the empty ontology [tCDK13]

Input A signature Σ
Output A k-ary CQ q_H such that $q_H \equiv_\varnothing q_T$

$q_H :=$ initial-CQ$(\Sigma, \varnothing)$
while the equivalence query "$q_H \equiv_\varnothing q_T$?" returns a counterexample $(\mathcal{A}, \overline{a})$ **do**
  $q_H(\overline{x} \otimes \overline{a}) :=$ minimize$_\varnothing(q_H \times \mathcal{A})$
end while
return q_H

We lift the restriction to queries of fixed arity in Section 5.4 by allowing learning algorithms to make CQ-equivalence queries, that is, equivalence queries where the hypothesis need not be a chordal symmetry-free CQ itself, but can be any CQ.

In Section 5.5, we look at $\mathcal{ELI}$ ontologies, which extend both $\mathcal{EL}^r$ and *DL-Lite*$_{\text{horn}}$ by permitting qualified existential restrictions and unrestricted inverse roles. We show that already the class of ELQs is not polynomial query learnable under $\mathcal{ELI}$ ontologies using membership queries and equivalence queries.

We briefly look at the learnability of query classes with disjunction and review related results in Section 5.6.

Finally, we conclude in Section 5.7 with a discussion about these results and possible future directions.

Related Publications

Sections 5.1 and 5.2 are based on [FJL22a]. Sections 5.3 to 5.5 are based on [FJL21a], but resolve an issue in the definition of symmetry-free CQ.

5.1 Updating Hypotheses with Counterexamples

The main question regarding the use of equivalence queries in learning algorithms is how counterexamples can be used to update the current hypothesis to be closer to the target query. One possible way for this is shown in Algorithm 5.1, which is a learning algorithm for CQs under the empty ontology. It is a special case of the learning algorithm for GAV schema mappings by ten Cate, Dalmau, and Kolaitis [tCDK13]. Like Algorithm 4.1 from Chapter 4, this algorithm starts with an initial query q_H^0 that implies q_T and then generalizes it step-by-step until q_H is equivalent to q_T. Since CQs in general do not possess frontiers of polynomial size, Algorithm 5.1 employs an equivalence query to obtain a counterexample from the teacher, with which it then updates q_H.

As the algorithm maintains $q_H \subseteq_\varnothing q_T$ at all times, there never exists a counterexample $(\mathcal{A}, \overline{a})$ such that $\mathcal{A}, \varnothing \not\models q_T(\overline{a})$ and $\mathcal{A}, \varnothing \models q_H(\overline{a})$. Instead, the teacher is always forced to return a counterexample $(\mathcal{A}, \overline{a})$ such that $\mathcal{A}, \varnothing \models q_T(\overline{a})$ and $\mathcal{A}, \varnothing \not\models q_H(\overline{a})$. In each iteration of its loop, Algorithm 5.1 uses this counterexample to construct the direct product $q_H \times (\mathcal{A}, \overline{a})$. To understand why this constitutes a suitable update of the hypothesis, it is best to view $(\mathcal{A}, \overline{a})$ as a CQ $q_{\mathcal{A}}(\overline{a})$, and to consider the properties of products stated in Lemma 3.3 together with the characterization of query implication in Lemma 3.7.

The properties of the counterexample $(\mathcal{A}, \overline{a})$ tell us that $q_{\mathcal{A}} \subseteq_\varnothing q_T$ and $q_{\mathcal{A}} \not\subseteq_\varnothing q_H$. It follows, if we view the product $q_H \times q_{\mathcal{A}}$ as a query with answer variables $\overline{x} \otimes \overline{a}$, that $q_H \times q_{\mathcal{A}} \subseteq_\varnothing q_T$ and $q_H \subseteq_\varnothing q_H \times q_{\mathcal{A}}$. Additionally, it must be that $q_H \times q_{\mathcal{A}} \not\subseteq_\varnothing q_H$ since $q_{\mathcal{A}} \subseteq_\varnothing q_H \times q_{\mathcal{A}}$. Hence, $q_H \times q_{\mathcal{A}}$ is a generalization of q_H, and all assignments to q_H during a run of Algorithm 5.1 form a generalization sequence towards q_T under $\varnothing$. As in Algorithm 4.1, the subroutine $\mathsf{minimize}_\varnothing$ assures $(q_T, \varnothing)$-minimality of the queries in the sequence, which bounds the number of loop iterations. We later show that such a bound also applies to CQs that are not rooted.

In the absence of an ontology, the initial CQ q_H^0 can easily be obtained. Given the signature Σ (and the required arity), the algorithm constructs

$$q_H^0(x, \dots, x) \leftarrow \bigwedge_{A \in \Sigma \cap \mathsf{N_C}} A(x) \wedge \bigwedge_{r \in \Sigma \cap \mathsf{N_R}} r(x, x).$$

By construction of q_H^0, $q_H^0 \subseteq_\varnothing q$ for all CQs q. Combining these arguments, one can show that Algorithm 5.1 learns CQs in polynomial time.

Proposition 5.3 ([tCDK13]). *Fix an arity $k \geq 0$. The class of all k-ary CQs is polynomial time learnable under the empty ontology using membership queries and equivalence queries.*

When we attempt to use Algorithm 5.1 for learning queries under non-empty ontologies, we run into two problems. The first problem is that if we learn ELIQs or ELQs, that the hypotheses used in equivalence queries must also be ELIQs or ELQs by definition of exact learning. This is not guaranteed by Algorithm 5.1, which uses CQs as hypotheses. Even when both q_H and $q_{\mathcal{A}}$ are ELIQs, their product need not be an ELIQ.

Example 5.4. Consider the ELIQs q and p in Figure 5.1. Their product $q \times p$ is not acyclic. This means that even if a counterexample is acyclic and hence corresponds to an ELIQ, the product $q_H \times q_{\mathcal{A}}$ need not be an ELIQ.

Fortunately, this problem can easily be addressed by using a subroutine like $\mathsf{extract}_{\mathrm{ELIQ}}$ defined in Section 4.5 to obtain a suitable ELIQ from $q_H \times q_{\mathcal{A}}$ using membership queries. The second problem concerns the effects of ontologies. If the ontology $\mathcal{O}$ is not empty, then it is not guaranteed that $q_H \times q_{\mathcal{A}} \subseteq_{\mathcal{O}} q_T$.

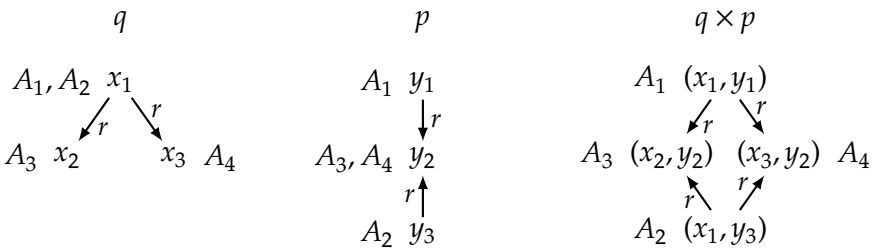

Figure 5.1: The ELIQs q and p, as well as their direct product $q \times p$.

Example 5.5. Consider the ontology $\mathcal{O} = \{A_1 \sqsubseteq B,\ A_2 \sqsubseteq B\}$ as well as the queries $q_H(x_0) \leftarrow A_1(x_0), q_{\mathcal{A}}(x_0) \leftarrow A_2(x_0)$, and $q_T(x_0) \leftarrow B(x_0)$. Then, $q_H \subseteq_{\mathcal{O}} q_T$ and $q_{\mathcal{A}} \subseteq_{\mathcal{O}} q_T$ but $q_H \times q_{\mathcal{A}}$ does not contain any atoms and therefore $q_H \times q_{\mathcal{A}} \not\subseteq_{\mathcal{O}} q_T$.

Hence, using the direct product to update hypotheses under ontologies does not result in a generalization sequence. We need an alternative way to update a hypothesis with a counterexample that takes the ontology into account. One approach to formalize this is to view the product $q_H \times q_{\mathcal{A}}$ as a *least general generalization* of both q_H and $q_{\mathcal{A}}$, and to consider this notion under ontologies.

Definition 5.6 (Least general generalization)**.** Let $\mathcal{Q}$ be a query class, $\mathcal{O}$ an ontology and p, q CQs of matching arity. A CQ $\widehat{q}$ is a $\mathcal{Q}$ *least general generalization* ($\mathcal{Q}$-LGG) of p and q under $\mathcal{O}$ if

1. $q \subseteq_{\mathcal{O}} \widehat{q}$;
2. $p \subseteq_{\mathcal{O}} \widehat{q}$;
3. $\widehat{q} \subseteq_{\mathcal{O}} q'$, for every $q' \in \mathcal{Q}$ with $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$.

Note the similarity of Definition 5.6 to *least common subsumers* of concepts, if $\mathcal{Q}$ is a class of queries that corresponds to DL concepts, like the class of all ELQs. The existence and computation of LCSs of ELQs under ontologies is well investigated, see for example [BST07; TZ13], and [JLW20]. The difference between Definition 5.6 and the LCS of two concepts is the requirement of the LCS to be from the same query class $\mathcal{Q}$. The LCS of two $\mathcal{EL}$ concepts must be an $\mathcal{EL}$ concept, but this is not the case for the ELQ-LGG. This means that LGGs may exist in situations where LCSs do not.

Also note that if $\mathcal{O} = \emptyset$, then a way to obtain a CQ least general generalization of two CQs is computing their direct product, as expected. If there is a way to compute CQ-LGGs under non-empty ontologies in polynomial time, then we could plug this into Algorithm 5.1 to obtain a learning algorithm for CQs under ontologies. However, as we see next, often it is not clear how such LGGs can be obtained.

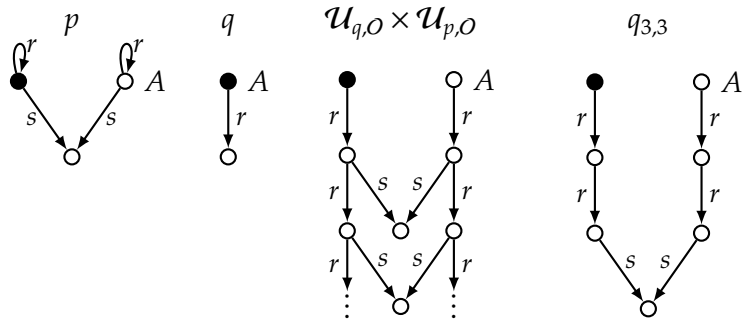

Figure 5.2: The queries p and q from Example 5.8 as well as the product of their universal models and the query $q_{3,3}$.

Let $\mathcal{O}$ be an ontology and $q(\bar{x}_1), p(\bar{x}_2), q'(\bar{x}_0)$ CQs that are satisfiable under $\mathcal{O}$ Recall that, by Lemma 3.5 and Lemma 3.7, $q \subseteq_{\mathcal{O}} q'$ if and only if $q'(\bar{x}_0) \to \mathcal{U}_{q,\mathcal{O}}, \bar{x}_1$. It follows then from Lemma 3.3 that $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$ if and only if $q'(\bar{x}_0) \to \mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}, \bar{x}_1 \otimes \bar{x}_2$. A natural choice for a least general generalization of q and p under $\mathcal{O}$ would therefore be $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ viewed as a query. Unfortunately, universal models are infinite in many cases, and therefore we cannot represent $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ as a finite CQ, let alone as a CQ of polynomial size.

Example 5.7. Consider the $\mathcal{EL}^r$ ontology

$$\mathcal{O} = \{A \equiv \exists r.A,\ A \equiv \exists s.A,\ B \equiv \exists r.B,\ B \equiv \exists s.B\}$$

and the CQs $q(x) \leftarrow A(x)$ and $p(x) \leftarrow B(x)$. The universal models $\mathcal{U}_{q,\mathcal{O}}$ and $\mathcal{U}_{p,\mathcal{O}}$ are both infinite binary trees, the former labeled with A, the latter with B. Their direct product $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ is an infinite tree that is not labeled with any concept name

Now suppose that $\hat{q}$ is a finite ELIQ-LGG of q and p under $\mathcal{O}$. As $q \subseteq_{\mathcal{O}} \hat{q}$ and $p \subseteq_{\mathcal{O}} \hat{q}$, $\hat{q}$ can only use the role names r and s. Consider all ELIQs q' that are $r-s$-paths. It holds that $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$, and therefore it must be the case that $\hat{q} \subseteq_{\mathcal{O}} q'$. Since this holds for an infinite number of $r-s$-paths and $\hat{q}$ is finite, it follows that $\hat{q}$ must contain cycles, contradicting that $q \subseteq_{\mathcal{O}} \hat{q}$.

Hence, no finite ELIQ-LGG of q and p under $\mathcal{O}$ exists, and therefore also no finite CQ-LGG.

Even if we know the size of the target query and are only interested in LGGs for queries of at most this size, then Example 5.7 suggests that any suitable LGG must be of exponential size, as there are an exponential number of different $r-s$-paths that need to be considered. Similar issues can also occur if we avoid qualified existential restrictions, and consider only *DL-Lite*$_{\text{core}}$ ontologies.

Example 5.8. Let $\mathcal{O} = \{\exists r^-.\top \sqsubseteq \exists r.\top,\ \exists r^-.\top \sqsubseteq \exists s.\top\}$ and consider the unary CQs

$$p(x) \leftarrow r(x,x) \wedge s(x,y) \wedge s(z,y) \wedge r(z,z) \wedge A(z) \quad \text{and} \quad q(x) \leftarrow A(x) \wedge r(x,y).$$

We argue that no finite ELIQ-LGG of p and q under $\mathcal{O}$ exists. Assume that there is a CQ $\widehat{q}(x)$ that is an ELIQ-LGG of p and q, and consider for all $n, m \geq 1$, the ELIQs

$$\begin{aligned} q_{n,m}(x_1) \leftarrow\ & r(x_1,x_2) \wedge \cdots \wedge r(x_{n-1},x_n) \wedge s(x_n,z) \wedge \\ & A(y_1) \wedge r(y_1,y_2) \wedge \cdots \wedge r(y_{m-1},y_m) \wedge s(y_m,z). \end{aligned}$$

The queries p, q and $q_{3,3}$ are depicted in Figure 5.2. Analyzing $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$, it can be verified that $p \subseteq_{\mathcal{O}} q_{n,m}$ and $q \subseteq_{\mathcal{O}} q_{n,m}$ if and only if $n = m$, thus $\widehat{q} \subseteq_{\mathcal{O}} q_{n,m}$ if and only if $n = m$. For all $i \geq 1$, let h_i be a homomorphism from $q_{i,i}$ to $\mathcal{U}_{\widehat{q},\mathcal{O}}$ that witnesses this. We distinguish cases.

- If there is an $i \geq 1$, such that h_i maps two variables $x_k, x_{k'} \in \mathsf{var}(q_{i,i})$ with $k \neq k'$ to the same element of $\mathcal{U}_{\widehat{q},\mathcal{O}}$, then a pumping argument shows that $\widehat{q} \subseteq_{\mathcal{O}} q_{j,i}$ for some $j > i$, a contradiction.

- Otherwise, by finiteness of $\widehat{q}$, there must be an $i \geq 1$ such that $h_i(z)$ is an element of $\mathcal{U}_{\widehat{q},\mathcal{O}}$ that was generated by an existential quantifier, a proper trace. Since the concept name A cannot occur on proper traces in $\mathcal{U}_{\widehat{q},\mathcal{O}}$, and $q_{i,i}$ is connected, the h_i-homomorphic image of $q_{i,i}$ must leave the proper traces of $\mathcal{U}_{\widehat{q},\mathcal{O}}$ again. Since the proper traces are tree-shaped, the image must enter and leave the proper traces at the same element of $\mathcal{U}_{\widehat{q},\mathcal{O}}$. Therefore, there are $n', m' \geq 0$ such that $\widehat{q} \subseteq_{\mathcal{O}} q'$ where

  $$\begin{aligned} q'(x_1) \leftarrow\ & r(x_1,x_2) \wedge \cdots \wedge r(x_{n'-1},z) \wedge \\ & A(y_1) \wedge r(y_1,y_2) \wedge \cdots \wedge r(y_{m'-1},z). \end{aligned}$$

  But $p \not\subseteq_{\mathcal{O}} q'$, contradicting that $\widehat{q}$ is an ELIQ-LGG of q and p under $\mathcal{O}$.

Example 5.7 and Example 5.8 leave little hope that we can use LGGs to update a ELIQ or CQ hypothesis with a counterexample under ontologies in polynomial time. In the following Section 5.2 and Section 5.3, we circumvent this in two different ways. In Section 5.2 we notice that we still obtain a generalization sequence towards q_T if we relax Point 3 of the definition of LGGs. For this, we define the notion of *guided generalization* and show that guided ELIQ-generalizations of ELIQs under *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ ontologies exist and can be constructed in polynomial time. This enables polynomial time learning of ELIQs under *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ ontologies. In Section 5.3 we instead focus on combinations of query class and ontology languages, for which we can replace the infinite $\mathcal{U}_{q_H,\mathcal{O}} \times \mathcal{U}_{q_{\mathcal{A}},\mathcal{O}}$ with a product of polynomially sized

compact models. The motivating example are ELQs under $\mathcal{EL}^r$ ontologies, for which the existence of such compact models is well known. We extend this to classes of ELIQs and CQs that exclude the problematic ELIQs used in Example 5.8. We show that these query classes are polynomial time learnable under $\mathcal{EL}^r$ ontologies.

5.2 Learning ELIQs under *DL-Lite*$_{\text{horn}}$ Ontologies

In Section 5.1 we have observed that finite ELIQ-LGGs of two CQs under *DL-Lite*$_{\text{core}}$ ontologies do not always exist. This means that in a learning algorithm for ELIQs we cannot use a product-like LGG construction to update an ELIQ hypothesis q_H with an equivalence query counterexample $q_{\mathcal{A}}$. However, if we recall the proof of Theorem 4.42, where we show that the hypotheses of Algorithm 4.1 form a generalization sequence towards the target query q_T, we see that for the updated hypothesis q'_H with $q_H \subseteq_{\mathcal{O}} q'_H \subseteq_{\mathcal{O}} q_T$, it is actually not necessary to demand that $q_{\mathcal{A}} \subseteq_{\mathcal{O}} q'_H$, it suffices to ensure that $q'_H \not\subseteq_{\mathcal{O}} q_H$. We capture this relaxed requirement in the following definition.

Definition 5.9 (Guided generalization)**.** Let $\mathcal{O}$ be an ontology and $\mathbf{Q}$ a query class, and p, q be CQs with $p \not\subseteq_{\mathcal{O}} q$. A CQ $\widehat{q}$ is a *p-guided $\mathbf{Q}$-generalization of q under $\mathcal{O}$* if

1. $q \subseteq_{\mathcal{O}} \widehat{q}$;
2. $\widehat{q} \not\subseteq_{\mathcal{O}} q$;
3. $\widehat{q} \subseteq_{\mathcal{O}} q'$, for every $q' \in \mathbf{Q}$ with $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$.

Note that assuming $p \not\subseteq_{\mathcal{O}} q$, every least general generalization of p and q under $\mathcal{O}$ is also a p-guided $\mathbf{Q}$-generalization of q under $\mathcal{O}$. The opposite is not true. Guided generalizations exist even when LGGs do not.

Example 5.10. Consider again the queries p and q and the ontology $\mathcal{O}$ from Example 5.8 that are depicted in Figure 5.2, and recall that there is no CQ-LGG of p and q under $\mathcal{O}$. In contrast, the ELIQ

$$\widehat{q}(x) \leftarrow r(x, y) \wedge r(x', y) \wedge A(x')$$

is a p-guided CQ-generalization of q under $\mathcal{O}$. This query also demonstrates that guided generalizations are an asymmetric notion, as $\widehat{q}$ is not a q-guided CQ-generalization of p under $\mathcal{O}$, since it does not satisfy Condition 1 of Definition 5.9.

Another consequence of this relaxation is that guided generalizations are not unique, while LGGs are.

Example 5.11. Consider $q(x) \leftarrow A(x) \wedge B(x) \wedge C(x)$ and $p(x) \leftarrow A(x)$. Then both $q_1(x) \leftarrow A(x)$ and $q_2(x) \leftarrow A(x) \wedge B(x)$ are p-guided CQ-generalizations of q under the empty ontology, and $q_1 \not\equiv_{\emptyset} q_2$.

It is also interesting to compare Definition 5.9, or more specifically guided ELIQ-generalizations, to elements of frontiers as defined in Definition 4.13. Note that Conditions 1 and 2 of Definition 5.9 correspond to Condition 1 of Definition 4.13, and Condition 3 of Definition 5.9 corresponds to Condition 2 of Definition 4.13. Intuitively, we can view the query p as a *guide* that helps us select a suitable query from a CQ-frontier of q, which is useful when q has an exponentially large or infinite CQ-frontier. Indeed, the query $\widehat{q}$ from Example 5.10 is an element of an ELIQ-frontier of q under $\mathcal{O}$.

Constructing Guided ELIQ-generalizations

We now show that p-guided ELIQ-generalizations of an ELIQ q always exist and can be computed in polynomial time when q if $(q, \mathcal{O})$-minimal, even under *DL-Lite*$_{\text{horn}}$ ontologies that use conjunctions. In fact, this is also the case under *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ ontologies $\mathcal{O}$ that are subject to the same restriction on functionality constraints as *DL-Lite*$^{\mathcal{HF}^-}_{\text{core}}$ ontologies in Chapter 4, that is, if a $\exists R.\top$ occurs on the right side of a concept inclusion in $\mathcal{O}$, then $\mathsf{func}(R^-) \notin \mathcal{O}$. We call this ontology language *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$.

For this, we again assume that $\mathcal{O}$ is in normal form, which is not a crucial assumption by Lemma 4.8. By inspecting the definition of the universal model of *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ ontologies, we find that we can safely adopt a restriction on the set of all traces [Bot+16]. Specifically, we restrict the domain of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to only contain traces $aR_1M_1 \cdots R_nM_n$ with $R_{i+1} \neq R_i^-$ for $1 \leq i < n$. Then, Lemma 3.5 still holds for *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ ontologies (but not for *DL-Lite*$^{\mathcal{H}}_{\text{core}}$ or $\mathcal{ELI}$ ontologies).

By adopting the above restriction, we can show the following property of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$, which in turn allows us to show that the construction of guided generalizations completes in polynomial time.

Lemma 5.12. *Let $\mathcal{O}$ be a DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontology in normal form, $\mathcal{A}$ an ABox that is satisfiable under $\mathcal{O}$, and $U = \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}} \setminus \mathsf{ind}(\mathcal{A})$. Then, for every role R, $R^{\mathcal{U}_{\mathcal{A},\mathcal{O}}} \cap U^2$ is a partial function.*

Note that universal models of *DL-Lite*$^{\mathcal{H}}_{\text{core}}$ or $\mathcal{EL}$ ontologies inherently do not have this property, as concept inclusions of the form $A \sqsubseteq \exists r.B_1$, $A \sqsubseteq \exists r.B_2$ require two different r successors for a model to be universal.

Let $q(x_1), p(x_2)$ be ELIQs and $\mathcal{O}$ a *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ ontology in normal form such that q and p are satisfiable under $\mathcal{O}$, q is $(q, \mathcal{O})$-minimal and $p \not\subseteq_{\mathcal{O}} q$. We construct a p-guided ELIQ-generalization $\widehat{q}$ of q under $\mathcal{O}$ in three steps.

1. Start with the query
$$\widehat{q}((x_1, x_2)) \leftarrow \bigwedge_{(x_1,x_2)\in A^{\mathcal{U}_{q,\mathcal{O}}\times\mathcal{U}_{p,\mathcal{O}}}} A((x_1, x_2)),$$
that is, the restriction of $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ to the element (x_1, x_2) viewed as a query with answer variable (x_1, x_2).

2. Then, extend $\widehat{q}$ by exhaustively applying the rule (A1) below.

   (A1) For every $(z, t) \in \mathsf{var}(\widehat{q})$ with $z \in \mathsf{var}(q)$ and $t \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$, every atom $R(z, z')$ in q, and every $(t, t') \in R^{\mathcal{U}_{p,\mathcal{O}}}$, add the atom $R((z, t), (z', t'))$, and all atoms $A((z', t'))$ such that $z' \in A^{\mathcal{U}_{q,\mathcal{O}}}$ and $t' \in A^{\mathcal{U}_{p,\mathcal{O}}}$.

3. Finally, complete $\widehat{q}$ by exhaustively applying the rule (A2) below.

   (A2) For every $(z, t) \in \mathsf{var}(\widehat{q})$ with $z \in \mathsf{var}(q)$ and $t \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$ and every role R such that $z \rightsquigarrow^{R}_{q,\mathcal{O}} M$ for some M add the atoms
$$R((z, t), \hat{z}), R(z', \hat{z})$$
with $\hat{z}$ a fresh variable, and add a copy q' of q in which the copy of z is z'.

To understand this construction, recall that $\mathcal{U}_{q,\mathcal{O}}\times\mathcal{U}_{p,\mathcal{O}}$, when viewed as an infinitary CQ, may serve as a CQ-LGG of p and q. Intuitively, the above construction may be viewed as producing an approximation of this product from below, in the sense that $\widehat{q} \subseteq_{\mathcal{O}} \mathcal{U}_{q,\mathcal{O}}\times\mathcal{U}_{p,\mathcal{O}}$ if we consider $\mathcal{U}_{q,\mathcal{O}}\times\mathcal{U}_{p,\mathcal{O}}$ to be restricted to all traces that are reachable from (x_1, x_2). After applying Rule (A1) exhaustively, we have constructed exactly the restriction of the product $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ to the elements (t_1, t_2) that are reachable from the element (x_1, x_2) and satisfy $t_1 \in \mathsf{var}(q)$. We show that this is a finite structure and even of polynomial size, which is essentially due to Lemma 5.12. What is missing is the infinite part of $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ determined by elements (t_1, t_2) where t_1 is a proper trace. The application of Rule (A2) approximates this part by adding atoms and copies of q corresponding to traces of length one in $\mathcal{U}_{q,\mathcal{O}}$.

Example 5.13. Consider the ELIQs q and p displayed in Figure 5.3 as well as the *DL-Lite*$_{\text{horn}}$ ontology $\mathcal{O} = \{A \sqsubseteq \exists s.\top\}$. The steps of computing a p-guided ELIQ-generalization of q under $\mathcal{O}$ are the queries $\widehat{q}_1, \widehat{q}_2, \widehat{q}_3$ displayed in Figure 5.3.

The query $\widehat{q}_1$ is the result of Step 1. Exhaustive application of Rule (A1) then yields $\widehat{q}_2$. Applying Rule (A2) once at the root then yields $\widehat{q}_3$. No other applications of Rule (A2) are possible, and $\widehat{q}_3$ indeed is a p-guided ELIQ-generalization of q. Note that there is a homomorphism from $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$ to $\widehat{q}_3$ and therefore also to $\mathcal{U}_{\widehat{q}_3,\mathcal{O}}$, demonstrating that $\widehat{q}_3 \subseteq_{\mathcal{O}} \mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$.

We show that this construction, especially Step 2, always terminates after polynomially many steps by using Lemma 5.12.

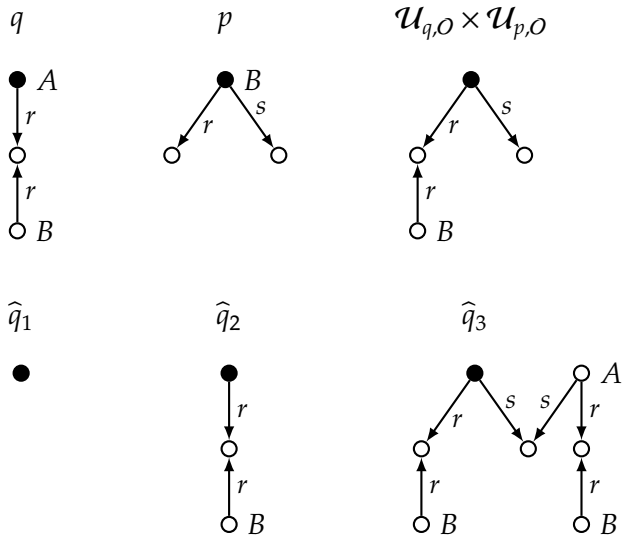

Figure 5.3: The three steps $\widehat{q}_1$, $\widehat{q}_2$, $\widehat{q}_3$ of the construction of a p-guided ELIQ-generalization of q under $\mathcal{O} = \{A \sqsubseteq \exists s.\top\}$.

Lemma 5.14. *The computation of $\widehat{q}$ terminates after polynomially many steps.*

Proof. The initial $\widehat{q}$ created in Step 1 can be computed in polynomial time, since it consists of a single variable and reasoning in *DL-Lite*$^{\mathcal{F}^-}_{\text{horn}}$ is possible in polynomial time. We consider Rule (A2) first. Let the result of Step 2 have domain size N. Then, (A2) is applied at most $N \cdot n_r$ times, where n_r denotes the number of roles in $\mathcal{O}$. Moreover, for each application, we only add two atoms and a copy of q. Thus, to show that the overall construction finishes in polynomial time, it suffices to show that Step 2 finishes in polynomial time.

For the analysis of the Rule (A1), observe that, by definition, Rule (A1) computes an initial fragment of the product $\mathcal{U}_{q,\mathcal{O}} \times \mathcal{U}_{p,\mathcal{O}}$. Thus, the rule creates at most $\|\mathcal{O}\| \cdot |\mathsf{var}(q)| \cdot |\mathsf{var}(p)|$ atoms over variables (x, y) with $x \in \mathsf{var}(q)$ and $y \in \mathsf{var}(p)$. The remaining rule applications can be structured into labeled trees T_{xy}, for each $(x, y) \in \mathsf{var}(q) \times \mathsf{var}(p)$, as follows:

- the root ε of T_{xy} is labeled with $\lambda(\varepsilon) = (x, y)$;
- if some node n is labeled with $\lambda(n) = (z, t)$ and Rule (A1) is applied to some $R(z, z') \in q$ and $(t, t') \in R^{\mathcal{U}_{p,\mathcal{O}}}$, then n has a successor n' with $\lambda(n') = (z', t')$; we additionally associate with n' another label $\rho(n') = R(z, z')$.

It suffices to bound the sizes of each tree T_{xy} by a polynomial in the input. For this, in turn, it suffices to show that there are no two nodes n_1, n_2 in T_{xy} such that $n_1 \neq n_2$,

$\lambda(n_1) = (z_1, t_1)$, $\lambda(n_2) = (z_2, t_2)$, and $z_1 = z_2$. We show this by contradiction.

Suppose there are $n_1 \neq n_2$ in T_{xy} such that $\lambda(n_1) = (z_1, t_1)$, $\lambda(n_2) = (z_2, t_2)$, and $z_1 = z_2$. Consider the unique shortest path from n_1 to n_2 in T_{xy} and let n be the unique node closest to the root on this path, that is, the path $w_0 \dots w_k$ from n_1 to n *goes up* in the tree and the path $v_0 \dots v_m$ from n to n_2 *goes down*. Consider the following sequence $\alpha_0, \dots, \alpha_{k+m-1}$ of facts:

(a) for $0 \leq i < k$, let α_i be the atom $R^-(z, z')$ when $\rho(w_i) = R(z', z)$;

(b) for $0 < i \leq m$, let $\alpha_{k+i-1} = \rho(v_i)$.

By definition of the Rule (A1) and the resulting definition of T_{xy}, the sequence $\alpha_0, \dots, \alpha_{k+m-1}$ is a path from z_1 to z_2 in q. Since $z_1 = z_2$ and q is acyclic, there has to be some i such that $\alpha_i = R(z, z')$ and $\alpha_{i+1} = R^-(z', z)$, for some role R. We distinguish cases on where α_i and α_{i+1} were defined in (a) or in (b) above.

Suppose first that both were defined in (a) and consider the nodes w_i, w_{i+1}. By definition of α_i, α_{i+1}:

- $\rho(w_i) = R^-(z, z')$ and $\rho(w_{i+1}) = R(z', z)$,
- $\lambda(w_i) = (z', t_1)$, for some t_1, and $\lambda(w_{i+1}) = (z, t_2)$, for some t_2.

Note that $\rho(w_i)$ and $\rho(w_{i+1})$ refer to the same atom. Let $(t, t_2) \in R^{\mathcal{U}_{p,\mathcal{O}}}$ be the pair such that w_{i+1} was added to T_{xy} via an application of Rule (A1) to $(z', t) \in \mathsf{var}(\widehat{q})$, $R(z', z) \in q$ and $(t, t_2) \in R^{\mathcal{U}_{p,\mathcal{O}}}$. By Lemma 5.12, R^- is a partial function when restricted to the domain $\Delta^{\mathcal{U}_{p,\mathcal{O}}} \setminus \mathsf{var}(p)$, and thus t_2 has no other R^--neighbor than t and thus $t_1 = t$. But then Rule (A1) is not applicable to $(z, t_2) \in \mathsf{var}(\widehat{q})$, $R^-(z, z') \in q$, and $(t_2, t_1) = (t_2, t) \in (R^-)^{\mathcal{U}_{p,\mathcal{O}}}$ since $R((z, t_2), (z', t_1)) = R((z, t_2), (z', t))$ is already present in $\widehat{q}$, a contradiction.

In the other two cases, where α_i, α_{i+1} were both defined in (b) or α_i was defined in (a) and α_{i+1} was defined in (b), a contradiction is derived analogously. □

Next, we show that the construction indeed yields a guided ELIQ-generalization. This relies heavily on the restriction on functionality constraints, as otherwise the atoms introduced by Rule (A2) may violate a functionality constraint $\mathsf{func}(R^-)$.

Lemma 5.15. *$\widehat{q}$ is satisfiable under $\mathcal{O}$ and is a p-guided ELIQ-generalization of q under $\mathcal{O}$.*

Proof. We show that $\widehat{q}$ satisfies Conditions 1 to 3 of Definition 5.9 and that $\widehat{q}$ is satisfiable under $\mathcal{O}$. For the proof, it is convenient to define a mapping g as follows:

- $g(z, t) = z$ for every $(z, t) \in \mathsf{var}(\widehat{q})$ with $z \in \mathsf{var}(q)$ and $t \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$;

- $g(\hat{z}) = zRM$, for every variable $\hat{z}$ introduced by Rule (A2) applied to the element (z, t), the role R and the set M;
- $g(x') = x$, for every copy x' of some variable x in q introduced by Rule (A2).

Observe that g is a homomorphism from $\widehat{q}$ to $\mathcal{U}_{q,\mathcal{O}}$ with $g(x_1, x_2) = x_1$, and thus $q \subseteq_{\mathcal{O}} \widehat{q}$. Hence, Condition 1 holds.

Satisfiability of $\widehat{q}$ under $\mathcal{O}$ follows from the facts that q is satisfiable under $\mathcal{O}$, that the map g defined above is a homomorphism from $\widehat{q}$ to $\mathcal{U}_{q,\mathcal{O}}$, and that since q satisfies all functionality assertions in $\mathcal{O}$, by construction so does $\widehat{q}$. For the latter, it is important that $\mathcal{O}$ is formulated in DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ rather than in DL-Lite$^{\mathcal{F}}_{\text{horn}}$. In particular, this ensures that when Rule (A2) is applied to a role R, it is not inverse functional.

For Condition 2, suppose to the contrary of what we have to show that $\widehat{q} \subseteq_{\mathcal{O}} q$. Since $\widehat{q}$ is satisfiable, we can fix a homomorphism h from q to $\mathcal{U}_{\widehat{q},\mathcal{O}}$ with $h(x_1) = (x_1, x_2)$. By Lemma 3.8, there is an extension of the homomorphism g to a homomorphism g' from $\mathcal{U}_{\widehat{q},\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$ with $g'(x_1, x_2) = x_1$. Then, the composition of h and g' is a homomorphism from q to $\mathcal{U}_{q,\mathcal{O}}$. Lemma 4.21 implies that the variables $\hat{z}$ introduced by Rule (A2) are not in the image of h. Indeed, if $h(x) = \hat{z}$ for some $x \in \mathsf{var}(q)$, then $g'(h(x)) \notin \mathsf{var}(q)$ takes the shape zRM, in contradiction to Lemma 4.21. Since q is rooted, all $h(x)$ take the shape (z, t) for some $z \in \mathsf{var}(q)$ and some trace t in $\mathcal{U}_{p,\mathcal{O}}$. Consider the projection h' of h to its second component, that is,

$$\text{for all } x \in \mathsf{var}(q) \text{ set } h'(x) = t \text{ where } h(x) = (z, t).$$

It is routine to show that h' is a homomorphism from q to $\mathcal{U}_{p,\mathcal{O}}$ with $h'(x_1) = x_2$, and thus $p \subseteq_{\mathcal{O}} q$, a contradiction.

For Condition 3, let $q'(x_0)$ be any ELIQ with $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$. We can fix a homomorphism h_p from q' to $\mathcal{U}_{p,\mathcal{O}}$ with $h_p(x_0) = x_2$ and a homomorphism h_q from q' to $\mathcal{U}_{q,\mathcal{O}}$ with $h_q(x_0) = x_1$. Based on h_q and h_p, we iteratively define a map h from q' to $\mathcal{U}_{\widehat{q},\mathcal{O}}$. We start by setting $h(x_0) = (h_q(x_0), h_p(x_0)) = (x_1, x_2)$. Now extend h iteratively by selecting an atom $R(x, x') \in q'$ such that $h(x) = (z, t)$ is defined, $z \in \mathsf{var}(q)$, and $h(x')$ is undefined. Note that $z' = h_q(x')$ satisfies $(z, z') \in R^{\mathcal{U}_{q,\mathcal{O}}}$ since h_q is a homomorphism. Similarly, $t' = h_p(x')$ satisfies $(t, t') \in R^{\mathcal{U}_{p,\mathcal{O}}}$. We distinguish cases.

1. Suppose first that $z' \in \mathsf{var}(q)$. Then, Rule (A1) is applicable to $R(z, z')$, (t, t'), and there is a $R((z, t), (z', t')) \in \widehat{q}$. Set $h(x') = (z', t')$.
2. Otherwise, $z' \notin \mathsf{var}(q)$. Since $z \in \mathsf{var}(q)$, z' takes the form zRM for some M and thus $z \rightsquigarrow^{R}_{q,\mathcal{O}} M$ for that M. Then Rule (A2) is applicable to (z, t), R and M. Let $\hat{z}$ be the variable introduced in Rule (A2). Using the definition of the

universal model, one can show that there is a homomorphism f from $\mathcal{U}_{q,\mathcal{O}}$ to $\mathcal{U}_{\widehat{q},\mathcal{O}}$ which maps zRM to $\hat{z}$ and q to the copy of q that was added to $\widehat{q}$ in this application of Rule (A2). We set

$$h(x'') = f(h_q(x''))$$

for every node x'' in the subtree rooted at x'.

It remains to argue that h is a homomorphism from q' to $\mathcal{U}_{\widehat{q},\mathcal{O}}$ with $h(x_0) = (x_1, x_2)$, and thus $\widehat{q} \subseteq_{\mathcal{O}} q'$. To see this, first, let $A(x) \in q'$.

- If $h(x)$ was defined in Case 1 above, then $h(x) = (z, t)$ for $z = h_q(x) \in \mathsf{var}(q)$ and $t = h_p(x) \in \Delta^{\mathcal{U}_{p,\mathcal{O}}}$. Since both h_q and h_p are homomorphisms, both $z \in A^{\mathcal{U}_{q,\mathcal{O}}}$ and $t \in A^{\mathcal{U}_{p,\mathcal{O}}}$. Thus, since (z, t) was created by Rule (A1), $A((z, t)) = A(h(x)) \in \widehat{q}$.

- If $h(x)$ was defined in Case 2 above, then $h(x) = f(h_q(x))$ where f is a homomorphism from q to $\mathcal{U}_{\widehat{q},\mathcal{O}}$. Since, additionally, h_q is a homomorphism, it follows that $A(h(x)) \in \mathcal{U}_{\widehat{q},\mathcal{O}}$.

Suppose now $R(x, y) \in q'$ and $R(x, y)$ is directed away from the root x_0 in q'.

- If both $h(x)$ and $h(y)$ were defined in Case 1, then Rule (A1) created the atom $R((h_q(x), h_p(x)), (h_q(y), h_p(y))) \in \widehat{q}$ and thus $R(h(x), h(y)) \in \widehat{q}$.

- If both $h(x)$ and $h(y)$ were defined in Case 2, then $h(x) = f(h_q(x))$ and $h(y) = f(h_q(x))$ for some homomorphism f from q to $\mathcal{U}_{\widehat{q},\mathcal{O}}$. Since, additionally, h_q is a homomorphism, it follows that $(h(x), h(y)) \in R^{\mathcal{U}_{\widehat{q},\mathcal{O}}}$.

- If $h(x)$ was defined in Case 1 and $h(y)$ was defined in Step 2, then $h(x) = (h_q(x), h_p(x)) = (z, t)$ and $h(y) = f(h_q(y)) = \hat{z}$ for the element $\hat{z}$ that was introduced in the application of Rule (A2) to (z, t) that defined $h(y)$. Rule (A2) additionally implies that $R((z, t), \hat{z}) \in \widehat{q}$, and thus $(h(x), h(y)) \in R^{\mathcal{U}_{\widehat{q},\mathcal{O}}}$.

- The case that $h(x)$ was defined in Case 2, but $h(y)$ was defined in Case 1 is not possible since x is closer to the root than y, by assumption and the fact that h is defined from root to leaves in q'. □

Combining Lemma 5.14 and Lemma 5.15 we thus obtain the following.

Lemma 5.16. *Given a DL-Lite$^{\mathcal{F}-}_{\text{horn}}$ ontology $\mathcal{O}$ in normal form and ELIQs p, q such that p, q are satisfiable under $\mathcal{O}$, $p \not\subseteq_{\mathcal{O}} q$, and q is $(q, \mathcal{O})$-minimal, a p-guided ELIQ-generalization of q under $\mathcal{O}$ that is satisfiable under $\mathcal{O}$ can be computed in polynomial time.*

Algorithm 5.2: Learning algorithm for ELIQs under DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontologies

Input A signature Σ and a DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontology $\mathcal{O}$ in normal form
Output An ELIQ q_H such that $q_H \equiv_{\mathcal{O}} q_T$

$q^0_H :=$ initial-CQ$(\Sigma, \mathcal{O})$
$q_H :=$ extract$_{\text{ELIQ}}(\mathcal{O}, q^0_H)$
while the equivalence query "$q_H \equiv_{\mathcal{O}} q_T$?" returns a counterexample $(\mathcal{A}, a)$ **do**
  $q_D :=$ extract$_{\text{ELIQ}}(\mathcal{O}, q_{\mathcal{A}}(a))$
  $q'_H :=$ a q_D-guided ELIQ-generalization of q_H under $\mathcal{O}$
  $q_H :=$ extract$_{\text{ELIQ}}(\mathcal{O}, q'_H)$
end while
return q_H

We conjecture that $(q, \mathcal{O})$-minimality of an ELIQ q under an DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontology can also be achieved in polynomial time by extending techniques for answering ELIQs over DL-Lite$_{\text{core}}$ ontologies in polynomial time in [Bie+13] to DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontologies. However, this is not needed to use Lemma 5.16 as part of a learning algorithm, as $(q, \mathcal{O})$-minimality can be achieved in polynomial time through membership queries.

Learning ELIQs under DL-Lite$^{\mathcal{F}^-}_{\text{horn}}$ ontologies

Lemma 5.16 enables us to use guided generalizations as part of a polynomial time ELIQ learning algorithm to update the hypothesis with a counterexample. Such an algorithm is displayed as Algorithm 5.2. Algorithm 5.2 replaces the direct product used in Algorithm 5.1 with the construction of guided generalizations. As part of this replacement, two further adjustments are necessary. First, Lemma 5.16 expects both queries p and q to be ELIQs, however the counterexample $(\mathcal{A}, a)$ viewed as a query $q_{\mathcal{A}}$ may not be acyclic. To work around this, Algorithm 5.2 applies the subroutine extract$_{\text{ELIQ}}$ from Section 4.5 to the counterexample and, since $q_{\mathcal{A}} \not\subseteq_{\mathcal{O}} q_H$ or equivalently $\mathcal{A}, \mathcal{O} \not\models q_H(a)$, obtains an ELIQ q_D with $q_D \not\subseteq_{\mathcal{O}} q_H$. As shown in Section 4.5, extract$_{\text{ELIQ}}$ works in polynomial time, using a polynomial number of membership queries. Second, the same issue occurs after the construction of a guided ELIQ-generalization. The hypothesis used in an equivalence query must also be an ELIQ, but the guided ELIQ-generalization q'_H constructed using Lemma 5.16 may not be acyclic. Again, Algorithm 5.2 applies extract$_{\text{ELIQ}}$ to obtain a suitable ELIQ q_H with $q'_H \subseteq_{\mathcal{O}} q_H \subseteq_{\mathcal{O}} q_T$.

The initial CQ q^0_H can be obtained in the same way as described in Section 4.5, by

using

$$q_H^0(x_0) \leftarrow \bigwedge_{A \in \Sigma \cap \mathsf{N_C}} A(x_0) \wedge \bigwedge_{r \in \Sigma \cap \mathsf{N_R}} r(x_0, x_0)$$

or a single equivalence query, depending on the disjointness constraints in $\mathcal{O}$. To show that Algorithm 5.2 is indeed a polynomial time learning algorithm, we show that the sequence $q_1, q_2, \ldots$ of assignments to q_H forms a generalization sequence towards q_T under $\mathcal{O}$. Since every *DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ ontology is also an $\mathcal{ELIHF}_\bot$ ontology, Theorem 4.35 then implies termination of Algorithm 5.2 after polynomially many steps. This is the main result of this section.

Theorem 5.17. *ELIQs are polynomial time learnable under DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ *ontologies using both equivalence and membership queries.*

Proof. We will show that Algorithm 5.2 is a polynomial time learning algorithm for ELIQs under *DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ ontologies in normal form. It follows from Lemma 4.8 that this also implies polynomial time learnability under general *DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ ontologies. To show this, let $\mathcal{O}$ be an *DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ ontology in normal form and let $q_1, q_2, \ldots$ be the sequence of assignments to the variable q_H during a run of Algorithm 5.2. We show that this sequence is a generalization sequence towards q_T under $\mathcal{O}$. Since $\mathsf{extract}_{\text{ELIQ}}$ maintains satisfiability by Lemma 4.41, counterexamples are satisfiable under $\mathcal{O}$ and the construction of guided generalizations maintains satisfiability by Lemma 5.16, all q_i are satisfiable under $\mathcal{O}$.

We first show that $q_i \subseteq_{\mathcal{O}} q_T$ for all i by induction on i. In the induction start, recall that $q_1 = \mathsf{extract}_{\text{ELIQ}}(q_H^0)$. Since $q_H^0 \subseteq_{\mathcal{O}} q_T$, Lemma 4.41 implies that $q_1 \subseteq_{\mathcal{O}} q_T$. Now assume that $q_{i-1} \subseteq_{\mathcal{O}} q_T$. If the sequence does not end at q_{i-1}, then the equivalence query returned a counterexample $(\mathcal{A}, a)$ with $\mathcal{A}, \mathcal{O} \models q_T(a)$ and $\mathcal{A}, \mathcal{O} \not\models q_{i-1}(a)$. Then, $q_D = \mathsf{extract}_{\text{ELIQ}}(q_{\mathcal{A}}) \subseteq_{\mathcal{O}} q_T$ and $q_D \not\subseteq_{\mathcal{O}} q_{i-1}$, and q'_H is a q_D-guided ELIQ generalization of q_H under $\mathcal{O}$. Therefore, $q'_H \subseteq_{\mathcal{O}} q_T$ and $q_i = \mathsf{extract}_{\text{ELIQ}}(q'_H) \subseteq_{\mathcal{O}} q_T$ by Lemma 4.41, as required.

Next, we show that $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$ for all i. Again, if the sequence did not end at q_i, then the equivalence query for q_i returned a counterexample $(\mathcal{A}, a)$ and $q_{i+1} = \mathsf{extract}_{\text{ELIQ}}(q'_H)$ for q'_H a q_D-guided ELIQ-generalization of q_H under $\mathcal{O}$. Then, $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$ by Definition 5.9 and Lemma 4.41.

Additionally, all q_i are $(q_T, \mathcal{O})$-minimal as they are the result of the $\mathsf{extract}_{\text{ELIQ}}$ subroutine. Therefore, $q_1, q_2, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$ and each q_i is $(q_T, \mathcal{O})$-minimal. As every *DL-Lite*$_{\text{horn}}^{\mathcal{F}-}$ ontology is also a $\mathcal{ELIHF}_\bot$ ontology, Theorem 4.35 then implies that the length of the sequence is bound by a polynomial. Therefore, Algorithm 5.2 must terminate with a hypothesis equivalent to q_T after a polynomial number of iterations. □

In Chapter 4 we have shown with Theorem 4.5 that membership queries alone do not suffice to learn ELIQs under *DL-Lite*$_{\text{horn}}$ ontologies in polynomial time. Here, we have shown that membership queries and equivalence queries suffice. Later, in Section 5.5 we will see that extending *DL-Lite*$_{\text{horn}}$ by qualified existential restrictions (obtaining $\mathcal{ELI}$) makes polynomial time learning with membership and equivalence queries impossible.

It is unclear how far Theorem 5.17 can be extended beyond *DL-Lite*$^{\mathcal{F}-}_{\text{horn}}$ and ELIQs. The known lower bound, that we discuss later, only applies to $\mathcal{ELI}$ ontologies (Theorem 5.50). It is especially not clear whether guided generalizations of ELIQs under *DL-Lite*$^{\mathcal{F}}_{\text{horn}}$ or *DL-Lite*$^{\mathcal{H}}_{\text{horn}}$ exist, and whether they can be constructed in polynomial time. At least, we can see that the construction described in this section fails to produce guided generalizations of polynomial size for these two ontology languages.

Note that since guided generalizations are not unique, the construction of guided generalizations in this section is somewhat arbitrary, and that other constructions, with perhaps additional desirable properties, are possible. Many questions remain open in the area of generalizations. One of direct interest to learning queries is whether guided CQ-generalizations of CQs exist under ontologies. Unfortunately, the idea of the construction in this section cannot be extended to CQs.

For learning algorithms, it could also be useful to consider generalizations of Definition 5.9. Perhaps constructions exist of *sets of guided generalizations* that behave in a suitable way and can be used by learning algorithms like frontiers.

Next, we turn away from guided generalizations, and instead consider query classes and ontology languages for which we can use products of compact models in place of LGGs.

5.3 Learning under $\mathcal{EL}^r$ Ontologies

As discussed in Section 5.1, an LGG of CQs can be computed in polynomial time in the case without an ontology, but already simple queries fail to have finite CQ-LGGs under ontologies that use qualified existential restrictions (Example 5.7). In particular, the direct product of universal models has all the properties needed for LGGs but is usually infinite. In this section, we aim to learn queries under $\mathcal{EL}^r$ ontologies that contain qualified existential restrictions, by focusing on query classes $\mathcal{Q}$ that allow us to replace the direct product of universal models with a direct product of finite models.

We observe two properties of $\mathcal{EL}^r$ ontologies. First, $\mathcal{EL}^r$ ontologies do not contain any concept or role disjointness constraints, and thus every ABox and CQ is satisfiable under $\mathcal{EL}^r$ ontologies. Second, $\mathcal{EL}^r$ ontologies and ABoxes possess

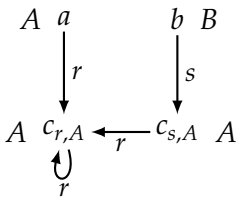

Figure 5.4: The compact model $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ of $\mathcal{A} = \{A(a), B(b)\}$ and $\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists s.\top,\ \exists s^-.\top \sqsubseteq A\}$.

ELQ-universal models of polynomial size, meaning that these models fulfill Point 3 of Lemma 3.5 but only for all ELQs. The main reason for this is that the use of inverse roles in $\mathcal{EL}^r$ is restricted, such that, for example, concept inclusions like the one in Example 4.36 cannot be expressed. ELQ-universal models of $\mathcal{EL}^r$ ontologies and ABoxes can be constructed as follows.

Let $\mathcal{O}$ be an $\mathcal{EL}^r$ ontology in normal form and $\mathcal{A}$ an ABox. The *compact universal model* $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ of $\mathcal{A}$ and $\mathcal{O}$ is obtained as follows. For every role name r, we use C_r to denote the conjunction over all A such that $\exists r^-.\top \sqsubseteq A \in \mathcal{O}$, and $\top$ if the conjunction is empty. We define the interpretation $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ by

$$
\begin{aligned}
\Delta^{\mathcal{C}_{\mathcal{A},\mathcal{O}}} &= \mathsf{ind}(\mathcal{A}) \cup \{c_{r,A} \mid r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{O}), A \in (\mathsf{sig}(\mathcal{O}) \cap \mathsf{N_C}) \cup \{\top\}\} \\
A^{\mathcal{C}_{\mathcal{A},\mathcal{O}}} &= \{a \in \mathsf{ind}(\mathcal{A}) \mid \mathcal{A},\mathcal{O} \models A(a)\} \cup \{c_{r,B} \mid \mathcal{O} \models B \sqcap C_r \sqsubseteq A\} \\
r^{\mathcal{C}_{\mathcal{A},\mathcal{O}}} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\
&\quad \{(a, c_{r,A}) \mid \mathcal{A},\mathcal{O} \models \exists r.A(a)\} \cup \\
&\quad \{(c_{s,A}, c_{r,B}) \mid \mathcal{O} \models A \sqcap C_s \sqsubseteq \exists r.B\}
\end{aligned}
$$

for all $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

Example 5.18. Consider the $\mathcal{EL}^r$ ontology $\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists s.\top,\ \exists s^-.\top \sqsubseteq A\}$ and the ABox $\mathcal{A} = \{A(a), B(b)\}$. The interpretation $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ is displayed in Figure 5.4, where the redundant elements $c_{s,\top}$ and $c_{r,\top}$ are left out. Note how infinite paths in $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ are represented by cycles in $\mathcal{C}_{\mathcal{A},\mathcal{O}}$, and how this makes $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ ELQ-universal but not ELIQ-universal or CQ-universal: Consider the ELIQ $q(x_0) \leftarrow r(x_0,x_1) \wedge r(x_2,x_1) \wedge r(x_3,x_2)$. It holds that $\mathcal{C}_{\mathcal{A},\mathcal{O}} \models q(a)$, but $\mathcal{A},\mathcal{O} \not\models q(a)$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}} \not\models q(a)$. In general, if $\mathcal{C}_{\mathcal{A},\mathcal{O}} \models q(a)$ and $\mathcal{A},\mathcal{O} \not\models q(a)$ for any CQ q, we refer to this as a *spurious match* of q in $\mathcal{C}_{\mathcal{A},\mathcal{O}}$.

Indeed, $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ is an ELQ-universal model of $\mathcal{A}$ and $\mathcal{O}$.

Lemma 5.19. *Let $\mathcal{A}$ be an ABox and $\mathcal{O}$ an $\mathcal{EL}^r$ ontology in normal form. Then,*

1. *$\mathcal{C}_{\mathcal{A},\mathcal{O}}$ is a model of $\mathcal{A}$ and $\mathcal{O}$;*

2. *for every ELQ q and $a \in \mathsf{ind}(\mathcal{A})$, $C_{\mathcal{A},\mathcal{O}} \models q(a)$ if and only if $\mathcal{A}, \mathcal{O} \models q(a)$.*

The proof of this is standard [LTW09]. We later provide a proof of universality of a different compact model construction (Lemma 5.28). For CQs q, we again write $C_{q,\mathcal{O}}$ instead of $C_{\mathcal{A}_q,\mathcal{O}}$.

Now, let $q(x_1)$ and $p(x_2)$ be ELQs and $\mathcal{O}$ an $\mathcal{EL}^r$ ontology in normal form. We are interested in the properties of $C_{q,\mathcal{O}} \times C_{p,\mathcal{O}}$ viewed as a CQ $\widehat{q}$ with the answer variable (x_1, x_2). First, note that $\widehat{q}$ is of size polynomial in $\|p\|$, $\|q\|$ and $\|\mathcal{O}\|$. Further, let q' be an ELQ such that $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$. It then follows from Lemma 5.19 and Lemma 3.7 that $C_{q,\mathcal{O}} \models q'(x_1)$ and $C_{p,\mathcal{O}} \models q'(x_2)$. By Lemma 3.3 it then follows that $C_{q,\mathcal{O}} \times C_{p,\mathcal{O}} \models q'(x_1, x_2)$. Since $\mathcal{EL}^r$ is monotone, it also follows that $C_{q,\mathcal{O}} \times C_{p,\mathcal{O}}, \mathcal{O} \models q'(x_1, x_2)$ and therefore $\widehat{q} \subseteq_{\mathcal{O}} q'$. This means that $\widehat{q}$ fulfills Point 3 of Definition 5.6 (LGGs). Note though that $\widehat{q}$ is not an LGG of q and p under $\mathcal{O}$, as it does not satisfy Point 1 and Point 2 of Definition 5.6. In fact, even $q \subseteq_{\mathcal{O}} C_{q,\mathcal{O}}$ does not hold for all ELQs q and $\mathcal{EL}^r$ ontologies $\mathcal{O}$.

Example 5.20. Consider the $\mathcal{EL}^r$ ontology $\mathcal{O} = \{A \sqsubseteq \exists r.A\}$ and the ELQ $q(x) \leftarrow A(x)$. Then, there is no homomorphism from $C_{q,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$ since $C_{q,\mathcal{O}}$ contains an element with a self-loop, and therefore $q \not\subseteq_{\mathcal{O}} C_{q,\mathcal{O}}$.

However, q, p, and $\widehat{q} = C_{q,\mathcal{O}} \times C_{p,\mathcal{O}}$ are of course not unrelated. In fact, for all ELQs q' it holds that $\widehat{q} \subseteq_{\mathcal{O}} q'$ implies $q \subseteq_{\mathcal{O}} q'$ and $p \subseteq_{\mathcal{O}} q'$, which suffices for obtaining an updated hypothesis. When receiving a counterexample $(\mathcal{A}, a)$, a learning algorithm can construct $\widehat{q} = C_{q_H,\mathcal{O}} \times C_{\mathcal{A},\mathcal{O}}$ to obtain a CQ that is a generalization of both q_H and $\mathcal{A}$ under $\mathcal{O}$. It is however not directly obvious that the algorithm makes progress towards q_T, since $q \subseteq_{\mathcal{O}} \widehat{q}$ does not necessarily hold. However, if the algorithm applies a subroutine like $\mathsf{extract}_{\mathrm{ELIQ}}$ to $\widehat{q}$ to obtain an ELQ from $\widehat{q}$, then this ELQ is a proper next element of a generalization sequence towards q_T. We show that this not only holds for ELQs, but also for larger query classes later in this section.

Compact Models for Larger Query Classes

We extend the idea of using $C_{q_H,\mathcal{O}} \times C_{\mathcal{A},\mathcal{O}}$ to update hypotheses to larger query classes. Unfortunately, this approach already fails for ELIQs. Consider the model $C_{\mathcal{A},\mathcal{O}}$ from Example 5.18 displayed in Figure 5.4 and the ELIQ $q(x_0) \leftarrow r(x_0, x_1) \wedge r(x_2, x_1) \wedge s(x_3, x_2) \wedge B(x_3)$. It holds that $C_{\mathcal{A},\mathcal{O}} \models q(a)$ but $\mathcal{A}, \mathcal{O} \not\models q(a)$, demonstrating that Point 2 of Lemma 5.19 does not hold for ELIQs instead of ELQs. The spurious match of q into $C_{\mathcal{A},\mathcal{O}}$ relies on the multiple r-predecessors of $c_{r,A}$ that are used to compactly represent multiple traces of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. Unfortunately, this issue cannot be avoided by using a more carefully defined finite model.

Example 5.21. Let $\mathcal{O} = \{\top \sqsubseteq \exists r.\top\}$ and $\mathcal{A} = \{A(a)\}$. The universal model $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ of $\mathcal{A}$ and $\mathcal{O}$ extends $\mathcal{A}$ with an infinite r-path consisting of traces $ar\emptyset, ar\emptyset r\emptyset, \ldots$. Any ELIQ-universal model also needs to contain such an infinite r-path, and thus the only chance to obtain a finite ELIQ-universal model $\mathcal{I}$ is to reuse elements on this path and create a cycle in $\mathcal{I}$. However, $\mathcal{I}$ cannot be ELIQ-universal then. Let h be a homomorphism from $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to $\mathcal{I}$. If h maps the trace of length n and the trace of length m with $n < m$ to the same element in $\mathcal{I}$, then $\mathcal{I} \models q(a)$ for the ELIQ

$$q(x_0) \leftarrow r(x_0, x_1) \land \cdots \land r(x_{m-1}, x_m) \land r(x_{m+1}, x_m) \land \cdots \land r(x_{m+n}, x_{m+n-1}) \land A(x_{m+n}),$$

but $\mathcal{A}, \mathcal{O} \not\models q(a)$.

Example 5.21 and the previous Example 5.8 indicate that atoms of the shape $r(y_1, x), r(y_2, x)$ for some role name r cause difficulties, as these allow ELIQs to see predecessors in the compact model $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ that do not exist in $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. We refer to these predecessors as *spurious*. The same issue with spurious predecessors also occurs with CQs that avoid inverse roles, but include cycles. Consider $\mathcal{A}$ and $\mathcal{O}$ as in Example 5.21. For the CQ $q(x_0) \leftarrow r(x_0, x_1) \land r(x_1, x_1)$, it holds that $\mathcal{C}_{\mathcal{A},\mathcal{O}} \models q(a)$ but $\mathcal{A}, \mathcal{O} \not\models q(a)$. We therefore refer to the cycle in $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ as a *spurious cycle*.

Since every finite model must have spurious predecessors, we can only circumvent these problems by restricting our attention to a subclass of ELIQs that does not allow this shape of atoms to occur. Similarly, we also consider a class of CQs that do not contain atoms of this shape and avoid large cycles, but still contains many useful CQs.

Let q be a CQ. A *path* of length n in q from $x \in \mathsf{var}(q)$ to $y \in \mathsf{var}(q)$ is a sequence $R_1(x_1, x_2), \ldots, R_n(x_n, x_{n+1}) \in q$ such that $x_1 = x$ and $x_{n+1} = y$. We say that a path is *simple* if $x_1, \ldots, x_{n+1}$ are distinct. We define the distance $\mathsf{dist}_q(x, y)$ of $x, y \in \mathsf{var}(q)$ to be the length of the shortest simple path from x to y, or to be ∞ if there is no path from x to y. A path is a *cycle of length n* if $x_1 = x_{n+1}$ and $x_2, \ldots, x_{n+1}$ are distinct. Note that this matches the definition of cycle in Section 4.5. A *chord* of a cycle is an atom $R(x_i, x_j)$ with $1 \leq i, j \leq n$ and $j \notin \{i-1 \bmod n+1,\ i,\ i+1 \bmod n+1\}$.

A *symmetry* in q consists of atoms $r(y_1, x), r(y_2, x)$ for some role name r and $y_1 \neq y_2$. We say that a symmetry is *safe*, if at least one of the following is true:

1. x is an answer variable,
2. at least one of the atoms occurs on a cycle in q,
3. q contains an atom $s(z, z)$ for some $z \in \{x, y_1, y_2\}$, or
4. q has only a single answer variable z, there is a path in q from z to x, and $\mathsf{dist}_q(z, x) + 1 = \mathsf{dist}_q(z, y_1) = \mathsf{dist}_q(z, y_2)$.

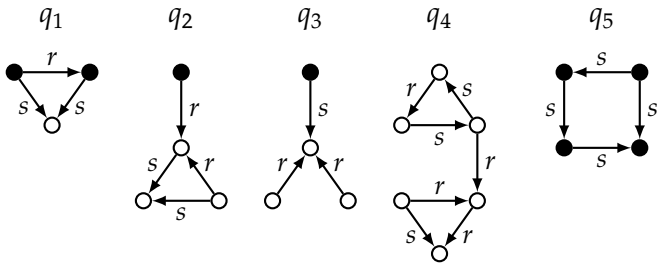

Figure 5.5: Examples of CQ^{csf} queries.

We denote with CQ^{csf} the class of CQs $q(\overline{x})$ that are

1. *chordal*: every cycle $R_1(x_1, x_2), \dots, R_{n-1}(x_{n-1}, x_1)$ in q of length at least four that contains at least one existential variable has a chord;
2. *symmetry-free*: every symmetry in q is safe.

Example 5.22. Consider the CQs q_1, q_2, q_3, q_4 and q_5 displayed in Figure 5.5. All of these queries are chordal and symmetry-free. The symmetries in q_1, q_2, q_4, q_5 are safe since they occur on a cycle. The symmetry in q_3 is safe since it fulfills Point 4. The queries q_1, q_2, q_3 and q_4 are chordal, since every cycle is of length smaller than four. The cycle in q_5 is chordal, since it does not contain any existential variable.

For ELIQs, we denote with ELIQ^{sf} the class of all ELIQs that are symmetry-free. If we view ELIQs as $\mathcal{ELI}$ concepts, then symmetry-free $\mathcal{ELI}$ concepts may not contain a subconcept of the shape $\exists r.(C_1 \sqcap \exists r^-.C_2)$. Due to the definition of symmetry and Point 4 of safety, all other subconcept shapes that involve role names and their inverses are allowed: $\exists r^-.(C_1 \sqcap \exists r.C_2)$, $\exists r^-.C_1 \sqcap \exists r.C_2$, $\exists r.C_1 \sqcap \exists r.C_2$, and $\exists r^-.C_1 \sqcap \exists r^-.C_2$. This restricted class of ELIQs was previously introduced by Jung, Lutz, and Wolter [JLW20] in the context of computing least common subsumers of $\mathcal{ELI}$ concepts under ontologies.

Note that every ELIQ^{sf} query is also a CQ^{csf} query, and that queries in CQ^{csf} need not be connected. Every CQ whose underlying graph is a clique or a k-tree (a maximal graph of treewidth k) is a CQ^{csf} query. Particularly, this means that many CQs that occur in practical applications are CQ^{csf}. For example, the three ontology-mediated querying benchmarks Fishmark [Bai+12], $\text{LUBM}^{\exists}$ [Lut+13], and NPD [Lan+15] contain 65 queries. Of those, 85 % are CQ^{csf} queries. Less than 5 % of the 65 queries are ELIQ^{sf} queries, mostly due to the existence of multiple answer variables.

We define a compact model of $\mathcal{EL}^r$ ontologies that is tailored towards the two properties of CQ^{csf} queries: chordality and being symmetry-free. The *3-compact*

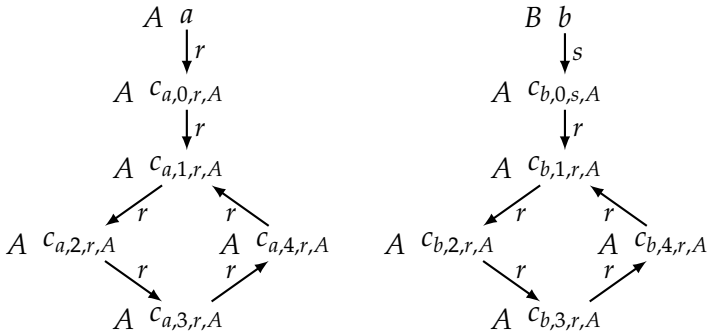

Figure 5.6: The 3-compact model $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ for $\mathcal{A} = \{A(a), B(b)\}$ and $\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists s.\top,\ \exists s^-.\top \sqsubseteq A\}$.

model $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ of an ABox $\mathcal{A}$ and an $\mathcal{EL}^r$ ontology $\mathcal{O}$ in normal form is defined as follows. The model uses the individual names from $\mathcal{A}$ as well as individual names of the form $c_{a,i,r,A}$ where $a \in \mathsf{ind}(\mathcal{A})$, $0 \leq i \leq 4$, r is a role name from $\mathcal{O}$, and $A \in (\mathsf{sig}(\mathcal{O}) \cap \mathsf{N_C}) \cup \{\top\}$. Let $i \oplus 1$ be short for $(i \bmod 4) + 1$. We define the interpretation $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ by

$$
\begin{aligned}
\Delta^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}} &= \mathsf{ind}(\mathcal{A}) \cup \\
&\quad \{c_{a,i,r,A} \mid a \in \mathsf{ind}(\mathcal{A}), 0 \leq i \leq 4, r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{O}), A \in (\mathsf{sig}(\mathcal{O}) \cap \mathsf{N_C}) \cup \{\top\}\} \\
A^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}} &= \{a \in \mathsf{ind}(\mathcal{A}) \mid \mathcal{A}, \mathcal{O} \vDash A(a)\} \cup \\
&\quad \{c_{a,i,r,B} \mid \mathcal{O} \vDash B \sqcap C_r \sqsubseteq A\} \\
r^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\
&\quad \{(a, c_{a,0,r,A}) \mid \mathcal{A}, \mathcal{O} \vDash \exists r.A(a)\} \cup \\
&\quad \{(c_{a,i,s,A}, c_{a,i\oplus 1,r,B}) \mid \mathcal{O} \vDash A \sqcap C_s \sqsubseteq \exists r.B\}
\end{aligned}
$$

for all $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$.

Example 5.23. Consider the same ABox $\mathcal{A} = \{A(a), B(b)\}$ and ontology $\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists s.\top,\ \exists s^-.\top \sqsubseteq \exists r.A\}$ as in Example 5.18. The interpretation $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ is displayed in Figure 5.6, or more precisely, the part of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ that is connected to $\mathcal{A}$. Again, redundant elements of the shape $c_{a,i,r,\top}$ and $c_{b,i,r,\top}$ are left out. Note that compared to the interpretation $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ displayed in Figure 5.4, all cycles have length 4 and that the elements attached to a are not connected to the elements attached to b.

We call $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ *3-compact* as it avoids all spurious cycles of length 3 or smaller, and thereby does not allow spurious matches of CQs that are chordal. The model also avoids spurious predecessors connected via different role names, or predecessors coming from different individuals in the ABox-part. The spurious predecessors

that remain are irrelevant for CQs that are symmetry-free. Note the existence of the elements $c_{a,i,r,A}$ with $i = 0$, which are not part of cycles. They exist to handle symmetries that are part of cycles, that could otherwise see the spurious predecessors of the elements of shape $c_{a,1,r,A}$.

Lemma 5.24. *Every cycle in $C^3_{\mathcal{A},\mathcal{O}}$ of length at most three consists only of individuals from* $\mathsf{ind}(\mathcal{A})$.

Proof. The statement is clear by construction of $C^3_{\mathcal{A},\mathcal{O}}$ for cycles of length 1. It is also clear for cycles of length 2 since for any pair of individuals of which at least one is of the form $c_{a,i,r,A}$, the ABox $C^3_{\mathcal{A},\mathcal{O}}$ contains at most one assertion that involves both of them.

Now for cycles of length 3. Assume to the contrary of what is to be shown that $C^3_{\mathcal{A},\mathcal{O}}$ contains a cycle of length 3 that contains an individual not from $\mathsf{ind}(\mathcal{A})$. First, assume that there is an individual $a \in \mathsf{ind}(\mathcal{A})$ on the cycle. Since all individuals of the form $c_{b,i,r,A}$ that are on the cycle are adjacent to a on the cycle, $b = a$ and $i = 0$ for all such $c_{b,i,r,A}$. This implies that a is the only individual from $a \in \mathsf{ind}(\mathcal{A})$ on the cycle. But then the cycle contains two distinct individuals of the form $c_{a,0,r,A}$ that are connected by an edge, which is never the case in $C^3_{\mathcal{A},\mathcal{O}}$.

Now assume that the cycle contains only individuals of the form $c_{b,i,r,A}$. Then all these individuals are connected in $C^3_{\mathcal{A},\mathcal{O}}$ by an edge. This is impossible due to the use of the index i in the construction of $C^3_{\mathcal{A},\mathcal{O}}$. □

There is a homomorphism from $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to $C^3_{\mathcal{A},\mathcal{O}}$ that is the identity on $\mathsf{ind}(\mathcal{A})$, but in general, there is no homomorphism from $C^3_{\mathcal{A},\mathcal{O}}$ to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. Nevertheless, $C^3_{\mathcal{A},\mathcal{O}}$ is $\mathrm{CQ}^{\mathrm{csf}}$-universal. Before showing this, we first describe how $C^3_{\mathcal{A},\mathcal{O}}$ relates to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ in terms of $\mathcal{EL}$ simulations.

Definition 5.25 ($\mathcal{EL}$ simulation)**.** An $\mathcal{EL}$ simulation from interpretation $\mathcal{I}_1$ to interpretation $\mathcal{I}_2$ is a relation $S \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ such that for all $(d_1, d_2) \in S$:

1. for all $A \in \mathsf{N_C}$: if $d_1 \in A^{\mathcal{I}_1}$, then $d_2 \in A^{\mathcal{I}_2}$;
2. for all $r \in \mathsf{N_R}$: if there is some $d_1' \in \Delta^{\mathcal{I}_1}$ with $(d_1, d_1') \in r^{\mathcal{I}_1}$, then there is a $d_2' \in \Delta^{\mathcal{I}_2}$ such that $(d_1', d_2') \in S$ and $(d_2, d_2') \in r^{\mathcal{I}_2}$.

We write $\mathcal{I}_1, d_1 \preceq_{\mathcal{EL}} \mathcal{I}_2, d_2$ if there exists an $\mathcal{EL}$ simulation S from $\mathcal{I}_1$ to $\mathcal{I}_2$ with $(d_1, d_2) \in S$.

$\mathcal{EL}$ simulations are similar to the $\mathcal{ELI}$ simulations defined in Definition 4.39, but they are not the same. For $\mathcal{EL}$ simulations, Point 2 only needs to be satisfied for role names, not inverse roles. The $\mathcal{EL}$ in the name of $\mathcal{EL}$ simulations refers to the fact that they are closely connected to $\mathcal{EL}$ concepts or ELQs.

Lemma 5.26. *Let $\mathcal{I}_1$, $\mathcal{I}_2$ be interpretations with $d_1 \in \Delta^{\mathcal{I}_1}$, $d_2 \in \Delta^{\mathcal{I}_2}$, and q an ELQ. If $\mathcal{I}_1, d_1 \preceq_{\mathcal{EL}} \mathcal{I}_2, d_2$, then $\mathcal{I}_1 \models q(d_1)$ implies $\mathcal{I}_2 \models q(d_2)$.*

The proof of this lemma is standard, see for example [LW10] and [CS01]. Using $\mathcal{EL}$ simulations, we can express the important relation of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ as follows.

Lemma 5.27. *Let $\mathcal{A}$ be an ABox and $\mathcal{O}$ an $\mathcal{EL}^r$ ontology in normal form. Further, let $c_{a,i,r,A} \in \Delta^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}}$ and $trM \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ with $A \in M$ (using the same role name r). Then, $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}, c_{a,i,r,A} \preceq_{\mathcal{EL}} \mathcal{U}_{\mathcal{A},\mathcal{O}}, trM$.*

Proof. We define the relation

$$S = \{(c_{a,i,r,A}, trM) \in \Delta^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}} \times \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}} \mid A \in M\}.$$

It suffices to show that S is an $\mathcal{EL}$ simulation from $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. Let $(c_{a,i,r,A}, trM) \in S$. If $c_{a,i,r,A} \in B^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}}$ for some concept name B, then $\mathcal{O} \models A \sqcap \exists r^-.\top \sqsubseteq B$. We aim to show that $trM \in B^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. If $t = a \in \mathsf{ind}(\mathcal{A})$, then $\mathcal{A}, \mathcal{O} \models \exists r. \bigsqcap M(a)$ and M must be maximal with this condition. Since $A \in M$ and $\mathcal{O} \models A \sqcap \exists r^-.\top \sqsubseteq B$, it follows that $B \in M$ and therefore, $trM \in B^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. If $t = t'r'M'$ is a proper trace, then $\mathcal{O} \models \bigsqcap M' \sqsubseteq \exists r. \bigsqcap M$ and M must be maximal with this condition. Since $A \in M$ and $\mathcal{O} \models A \sqcap \exists r^-.\top \sqsubseteq B$, it follows that $B \in M$.

If $(c_{a,i,r,A}, c_{b,j,r',B}) \in s^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}}$ for some role name s, then $r' = s$ by definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$. We aim to show that there is an $trMr'M' \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ such that $(trM, trMr'M') \in r'^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ and $(c_{b,j,r',B}, trMr'M') \in S$. From the definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ it follows that $\mathcal{O} \models A \sqcap C_r \sqsubseteq \exists r'.B$. Since, as argued above, $\emptyset \models \bigsqcap M \sqsubseteq A \sqcap C_r$, there is a set M' such that $M \rightsquigarrow^{r'}_{\mathcal{O}} M'$ and $B \in M'$. Therefore, there is a trace $trMr'M' \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ with $(trM, trMr'M') \in r'^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$ and $(c_{b,j,r',B}, trMr'M') \in S$. □

Most importantly, Lemma 5.27 can be used to show that $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}, a \preceq_{\mathcal{EL}} \mathcal{U}_{\mathcal{A},\mathcal{O}}, a$ for all $a \in \mathsf{ind}(\mathcal{A})$. Using the properties of $\mathrm{CQ}^{\mathrm{csf}}$ queries, we now show that $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ is a $\mathrm{CQ}^{\mathrm{csf}}$-universal model of $\mathcal{A}$ and $\mathcal{O}$.

Lemma 5.28. *Let $\mathcal{A}$ be an ABox and $\mathcal{O}$ an $\mathcal{EL}^r$ ontology in normal form. Then,*

1. *$\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ is a model of $\mathcal{A}$ and $\mathcal{O}$;*
2. *for every k-ary $q(\overline{x}) \in \mathrm{CQ}^{\mathrm{csf}}$ and $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$, $\mathcal{C}^3_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$ if and only if $\mathcal{A}, \mathcal{O} \models q(\overline{a})$.*

Proof. Point 1 follows from the definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$, details are omitted. For Point 2, let $q(\overline{x})$ be a k-ary $\mathrm{CQ}^{\mathrm{csf}}$ query and $\overline{a} \in \mathsf{ind}(\mathcal{A})^k$. We have to show that $\mathcal{C}^3_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$ if and only if $\mathcal{A}, \mathcal{O} \models q(\overline{a})$. For the first direction, assume that $\mathcal{A}, \mathcal{O} \models q(\overline{a})$. By

Lemma 3.5, it follows that $\mathcal{U}_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$. Since there is a homomorphism from $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ to $C^3_{\mathcal{A},\mathcal{O}}$ that is the identity on $\mathsf{ind}(\mathcal{A})$, $C^3_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$. For the second direction, assume that $C^3_{\mathcal{A},\mathcal{O}} \models q(\overline{a})$. Then, there is a homomorphism h from q to $C^3_{\mathcal{A},\mathcal{O}}$ with $h(\overline{x}) = \overline{a}$. In what follows, we construct a homomorphism g from q to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ with $g(\overline{x}) = \overline{a}$. Thus, $\mathcal{A}, \mathcal{O} \models q(\overline{a})$ as required.

To start the definition of g, set $g(x) = h(x)$ whenever $h(x) \in \mathsf{ind}(\mathcal{A})$. It follows from the construction of $C^3_{\mathcal{A},\mathcal{O}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ that g is a homomorphism from q restricted to the domain of g to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$.

Because of Lemma 5.24, if a variable x occurs on a cycle of length 1 or 2 in q, then $g(x)$ is now defined. We next define $g(x)$ for all variables x_0 that are on a cycle $R_0(x_0, x_1), R_1(x_1, x_2), R_2(x_2, x_0)$ of length 3 in q. Assume that $g(x_0)$ was not yet defined. It then follows from Lemma 5.24 that $h(x_1) = h(x_2) \in \mathsf{ind}(\mathcal{A})$, and thus $\mathcal{A}$ contains a reflexive R_1-cycle on $h(x_1)$, $R_0 = R_2^-$, and $h(x_0) \notin \mathsf{ind}(\mathcal{A})$. Let $h(x_1) = a$. By construction of $C^3_{\mathcal{A},\mathcal{O}}$, $h(x_0) = c_{a,0,r,A}$ for some A and where $r = R_0$ if R_0 is a role name and $r = R_2$ otherwise. Then there must be a set M with $A \in M$ such that $\mathcal{A}, \mathcal{O} \models \exists r. \bigsqcap M(a)$. If there is a $b \in \mathsf{ind}(\mathcal{A})$ with $r(a, b) \in \mathcal{A}$ and $\mathcal{A}, \mathcal{O} \models \bigsqcap M(b)$, then set $g(x_0) = b$. Otherwise, there is a trace $arM \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. Set $g(x_0) = arM$. After this extension, g is a homomorphism from the restriction of q to the (now extended) domain of g to $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. This is easily seen to be a consequence of the definition of the extension and of the construction of $C^3_{\mathcal{A},\mathcal{O}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}}$.

At this point, $g(x)$ is defined for all variables x that occur on a cycle in q. Assume that x is such a variable. If x is an answer variable, then $g(x)$ is clearly already defined. Otherwise, chordality of q implies that x also occurs on a cycle of length at most 3 and thus $g(x)$ has been defined above. It remains to define $g(x)$ for variables x that do not occur on a cycle.

We begin by mapping certain symmetries in q. Let $r(y_1, x), r(y_2, x)$ be a symmetry such that $g(x)$ is undefined and there is an atom $s(y_i, y_i)$ for some $i \in \{1, 2\}$. Then $g(y_i)$ must be defined, $h(y_i) \in \mathsf{ind}(\mathcal{A})$, and by definition of $C^3_{\mathcal{A},\mathcal{O}}$, $h(y_1) = h(y_2)$. Hence, $g(y_j)$ must be defined for $j \in \{1, 2\}$ and $h(x) = c_{h(y_i),0,r,A}$ for some concept name A. Then, as above, there must be a set M with $A \in M$ such that $\mathcal{A}, \mathcal{O} \models \exists r. \bigsqcap M(h(y_i))$. If there is a $b \in \mathsf{ind}(\mathcal{A})$ with $r(h(y_i), b) \in \mathcal{A}$ and $\mathcal{A}, \mathcal{O} \models \bigsqcap M(b)$, then set $g(x) = b$. Otherwise, there is a trace $h(y_i)rM \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. Set $g(x) = g(y_i)rM$.

We continue by mapping the rest of q. Let q' be the subquery of q consisting of all atoms that contain at least one variable x with $g(x)$ undefined at this point. We make four observations about q'.

1. Observe that no atom in q' is part of a cycle, as otherwise g would already be defined for both variables of this atom.

2. If $g(x)$ is defined for some variable x, then q' contains no atom $r(y, x)$.

To ascertain this, assume that there is such an atom. Then $g(y)$ must be undefined by choice of q', and $h(y) \notin \mathsf{ind}(\mathcal{A})$. Since $g(x)$ is defined, $h(x) \in \mathsf{ind}(\mathcal{A})$ or $h(x)$ is of the shape $c_{a,0,s,A}$. But by definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$, no $a \in \mathsf{ind}(\mathcal{A})$ and no element of shape $c_{a,0,s,A}$ has a predecessor that is not in $\mathsf{ind}(\mathcal{A})$, hence h cannot be a homomorphism, a contradiction.

3. If q' contains atoms $r_1(y_1, x), r_2(y_2, x)$, then $r_1 = r_2$.

   By the previous observation, $g(x)$ must be undefined, hence $h(x) \notin \mathsf{ind}(\mathcal{A})$. Therefore, $h(x) = c_{a,i,r,A}$ for some a, i, r, A. But by construction of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$, $c_{a,i,r,A}$ only has r-predecessors and since h is a homomorphism, $r_1 = r_2 = r$.

4. q' does not contain any symmetry.

   Assume that there is a symmetry $r(y_1, x), r(y_2, x) \in q$ with $y_1 \neq y_2$. Since q is symmetry-free, $r(y_1, x), r(y_2, x)$ must be a safe symmetry in q. By the previous observation, $g(x)$ cannot yet be defined. Hence, x is not an answer variable, no atom of the symmetry is part of a cycle, and there is no atom $s(x, x) \in q$. If there is an atom $s(y_1, y_1) \in q$ or $s(y_2, y_2) \in q$, then $g(x)$ was already defined earlier. Thus, $r(y_1, x), r(y_2, x)$ must be a safe symmetry for the reason that $\overline{x}$ consists of a single answer variable z with $\mathsf{dist}_q(z, x) + 1 = \mathsf{dist}_q(z, y_1) = \mathsf{dist}_q(z, y_2)$.

   This implies, that there is a simple path $R_1(x_1, x_2), \dots, R_n(x_n, x_{n+1})$ in q with $x_1 = z$, $x_{n+1} = x$ and $x_n \notin \{y_1, y_2\}$. By construction of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$, either $R_n = r$ or $R_n = s^-$ for some role name s. If $R_n = r$, then the atoms $R_n(x_n, x), r(y_1, x)$ form another symmetry. Since $g(x)$ is undefined, and $\mathsf{dist}_q(z, x_n) + 1 = \mathsf{dist}_q(z, x)$, this is not a safe symmetry, which contradicts that q is symmetry-free. Hence, R_n must be an inverse role s^- for some role name s. This means that if $h(x_{n+1}) = c_{a,i,r,A}$ for some a, i, and A, then $h(x_n) = c_{a,i\oplus 1,s,B}$.

   Consider how h maps the path $R_1(x_1, x_2), \dots, R_n(x_n, x_{n+1})$ into $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$. There must be an i such that $h(x_i) = a$ and $h(x_{i+1}) = c_{a,0,R_i,A}$ for some A and $a \in \mathsf{ind}(\mathcal{A})$ and $g(x_j)$ is undefined for every $j > i + 1$. By definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ it thus must be the case that there is some $j > i$ such that R_j is a role and $R_{j+1} = R_j^-$. This makes $R_j(x_j, x_{j+1}), R_{j+1}(x_{j+1}, x_{j+2})$ a symmetry in q. We argue that this symmetry is not safe, leading to a contradiction. First, since $g(x_{j+1})$ is undefined, x_{j+1} is not an answer variable, no atom occurs on a cycle and there is no atom $s(\hat{z}, \hat{z})$ for some $\hat{z} \in \{x, y_1, y_2\}$. Then, the path $R_1(x_1, x_2), \dots, R_j(x_j, x_{j+1})$ witnesses that $\mathsf{dist}_q(z, x_{j+1}) \neq \infty$ and that $\mathsf{dist}_q(z, x_j) < \mathsf{dist}_q(z, x_{j+1})$. Therefore, there cannot be a symmetry in q'.

Using these observations about q', we conclude that q' is a disjoint union of directed trees such that if $g(x)$ is defined for a variable x in q', then x is the root of a directed tree. We next extend g to the entirety of q by traversing the directed trees

in q' in a top-down fashion. The initial piece of g constructed so far is such that for all variables x, $h(x) = c_{a,i,r,A}$ implies that $g(x)$ is a proper trace of the form trM with $A \in M$ or a $b \in \mathsf{ind}(\mathcal{A})$ with $\mathcal{A}, \mathcal{O} \models A(b)$. We shall maintain this invariant during the extension of g. To extend g to all variables of q, repeatedly and exhaustively choose atoms $r(x, y) \in q'$ with $g(x)$ defined and $g(y)$ undefined. By the construction of g so far, $h(y) \notin \mathsf{ind}(\mathcal{A})$ and thus $h(y)$ has the form $c_{a,i,r,A}$. If $g(x) \in \mathsf{ind}(\mathcal{A})$, then $\mathcal{A}, \mathcal{O} \models \exists r. \bigsqcap M(g(x))$ for some set M with $A \in M$. If then there is a $b \in \mathsf{ind}(\mathcal{A})$ with $r(g(x), b) \in \mathcal{A}$ and $\mathcal{A}, \mathcal{O} \models \bigsqcap M(b)$, set $g(y) = b$. Otherwise, there is a trace $g(x)rM \in \Delta^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. Set $g(y) = g(x)rM$. If $g(x)$ is a proper trace, map $g(y)$ to a trace $g(x)rM$ that exists by Lemma 5.27.

It remains to verify that g is a homomorphism. If $h(x) \in \mathsf{ind}(\mathcal{A})$, then it follows immediately from the definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ that $(h(x), h(y)) \in r^{\mathcal{C}^3_{\mathcal{A},\mathcal{O}}}$ implies $(g(x), g(y)) \in r^{\mathcal{U}_{\mathcal{A},\mathcal{O}}}$. If $h(x) \notin \mathsf{ind}(\mathcal{A})$, we need to additionally invoke Lemma 5.27, applied to $h(x) = c_{a,i,r,A}$ and to $g(x) = trM$. Therefore, g satisfies all binary atoms in q' and thus in q. All unary atoms are satisfied, too, because of the invariant mentioned above and by definition of $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ and $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. □

Generalization Sequences of Unrestricted CQs

With $\mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ we have defined a suitable finite $\mathrm{CQ}^{\mathrm{csf}}$-universal model to update a hypothesis q_H with a counterexample $(\mathcal{A}, \overline{a})$ under an $\mathcal{EL}^r$ ontology. Constructing the direct product $\mathcal{C}^3_{q_H,\mathcal{O}} \times \mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ allows us to obtain a new hypothesis q'_H that generalizes q_H and further approaches the target query q_T. In Chapter 4, we used *generalization sequences* to argue that a sequence of such hypotheses must arrive at q_T after a polynomial number of steps, even under $\mathcal{ELIHF}_\bot$ ontologies. Unfortunately, Theorem 4.35 does not hold for $\mathrm{CQ}^{\mathrm{csf}}$ queries that are not rooted, as Example 4.36 demonstrates. For the purposes of this chapter, however, it suffices for sequences of hypotheses to be polynomially bounded under $\mathcal{EL}^r$ ontologies. We show that this is the case.

In Chapter 4 a central ingredient for achieving polynomial time learning was Lemma 4.20 which connects $(q_T, \mathcal{O})$-minimality to the image of homomorphisms. If q_T is not rooted, then Lemma 4.20 no longer holds.

Example 5.29. Consider the Boolean CQs $q_T() \leftarrow A(x)$, and $q() \leftarrow B(y)$ as well as the ontology $\mathcal{O} = \{B \sqsubseteq \exists r.A\}$. Then $q \subseteq_{\mathcal{O}} q_T$ and q is $(q_T, \mathcal{O})$-minimal, but $y \in \mathsf{var}(q)$ is not in the image of any homomorphism from q_T to $\mathcal{U}_{q,\mathcal{O}}$.

Fortunately, under $\mathcal{EL}^r$ ontologies, a weaker version of Lemma 4.20 holds even for unrestricted CQs, that is, also for CQs that are not rooted. This version suffices to bound the length of sequences of $(q_T, \mathcal{O})$-minimal queries. Let $\mathcal{O}$ be an $\mathcal{EL}^r$ ontology

in normal form, p, q CQs and h a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$. We define a function h^* from $\mathsf{var}(p)$ to $\mathsf{var}(q)$ by setting $h^*(y) = x$ for all $y \in \mathsf{var}(p)$ if $h(y)$ is a trace that starts with $x \in \mathsf{var}(q)$.

Lemma 5.30. *Let $\mathcal{O}$ be an $\mathcal{EL}^r$ ontology in normal form, $p(\overline{y})$ and $q(\overline{x})$ CQs such that q is $(p,\mathcal{O})$-minimal. If h is a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{y}) = \overline{x}$, then $\mathsf{var}(q) \subseteq \mathsf{img}(h^*)$.*

Proof. The proof is similar to the proof of Lemma 4.20.Let h be a homomorphism as required and assume for the sake of showing a contradiction that there is an $x \in \mathsf{var}(q)$ such that there is no $y \in \mathsf{var}(p)$ with $h^*(y) = x$. Let $q' = q^{-ox}$. We show that h is also a homomorphism from p to $\mathcal{U}_{q',\mathcal{O}}$ with $h(\overline{y}) = \overline{x}$, witnessing $q' \subseteq_{\mathcal{O}} p$ which contradicts that q is $(p,\mathcal{O})$-minimality.

First, note that by definition of h^* and choice of x, there is no variable $y \in \mathsf{var}(p)$ that is mapped by h to a trace starting with x in $\mathcal{U}_{q,\mathcal{O}}$. Let $x'RM$ be a trace of length one in $\mathcal{U}_{q,\mathcal{O}}$ with $x' \neq x$. By construction of $\mathcal{U}_{q,\mathcal{O}}$ and normal form of $\mathcal{O}$, the existence of $x'RM$ implies that there is a concept name A such that $\mathcal{A}_q, \mathcal{O} \models A(x')$ and $\mathcal{O} \models A \sqsubseteq \exists R.\top$. By construction of q' it follows that $\mathcal{A}_{q'}, \mathcal{O} \models A(x')$ and hence, $x'RM$ is also a trace in $\mathcal{U}_{q',\mathcal{O}}$. Therefore, all traces in $\mathcal{U}_{q,\mathcal{O}}$ that do not start with x' also occur in $\mathcal{U}_{q',\mathcal{O}}$ and h is a well-defined function from p to $\mathcal{U}_{q',\mathcal{O}}$.

Let $A(y)$ be a concept atom in p. Since $h(y) \in A^{\mathcal{U}_{q,\mathcal{O}}}$ and $h(y)$ is not a trace starting with x, the construction of q' implies that $h(y) \in A^{\mathcal{U}_{q',\mathcal{O}}}$. Let $r(y,y')$ be a role atom in p. Since $h(y)$ and $h(y')$ both are not traces starting with x, $(h(y), h(y')) \in r^{\mathcal{U}_{q,\mathcal{O}}}$ implies that $(h(y), h(y')) \in r^{\mathcal{U}_{q',\mathcal{O}}}$. Therefore, h is a homomorphism as required. □

From Lemma 5.30, we also directly obtain an equivalent of Lemma 4.21 for disconnected queries that will be useful later.

Lemma 5.31. *Let $\mathcal{O}$ be an $\mathcal{EL}^r$ ontology in normal form and $q(\overline{x}_0)$ a CQ that is $(q,\mathcal{O})$-minimal. For all homomorphisms h from q to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{x}_0) = \overline{x}_0$, $\mathsf{var}(q) = \mathsf{img}(h^*)$.*

Using Lemma 5.30 we can now show that under $\mathcal{EL}^r$ ontologies, even the length of generalization sequences of unrestricted CQs is bounded by a polynomial.

Theorem 5.32. *Let q_T be a CQ (that is possibly not rooted) and $\mathcal{O}$ an $\mathcal{EL}^r$ ontology in normal form, and let $q_1, q_2, \ldots$ be a generalization sequence towards q_T under $\mathcal{O}$. If all q_i are $(q_T,\mathcal{O})$-minimal, then the sequence has length at most $p(|\mathsf{var}(q_T)| + |\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$ for some fixed polynomial p.*

Proof. This proof is similar to the one of Theorem 4.35, but does additional steps to handle queries that are not rooted.

First, let $q_i(\overline{x}_i)$ be any element of the sequence. Since $q_i \subseteq_{\mathcal{O}} q_T$, there is a homomorphism h from q_T to $\mathcal{U}_{q_i,\mathcal{O}}$ with $h(\overline{x}) = \overline{x}_i$. Lemma 5.30 then implies that $\mathsf{var}(q_i) \subseteq \mathsf{img}(h^*)$. As $|\mathsf{img}(h^*)| \leq |\mathsf{var}(q_T)|$, it follows that $|\mathsf{var}(q_i)| \leq |\mathsf{var}(q_T)|$.

Since for all i, $q_i \subseteq_{\mathcal{O}} q_{i+1}$ by definition of generalization sequences, we fix homomorphisms h_i from q_{i+1} to $\mathcal{U}_{q_i,\mathcal{O}}$ with $h(\overline{x}_{i+1}) = \overline{x}_i$.

Assume that for some q_i, there is an $x \in \mathsf{var}(q_i)$ with $x \notin \mathsf{img}(h_i^*)$ and let h_i' be the extension of h_i to a homomorphism from $\mathcal{U}_{q_{i+1},\mathcal{O}}$ to $\mathcal{U}_{q_i,\mathcal{O}}$ with $h_i'(\overline{x}_{i+1}) = \overline{x}_i$ which exists by Lemma 3.8. Analogously to the proof of Theorem 4.35, we construct a homomorphism h_i'' from $\mathcal{U}_{q_{i+1},\mathcal{O}}$ to $\mathcal{U}_{q_i^- \mathcal{O}^x,\mathcal{O}}$ with $h_i''(\overline{x}_{i+1}) = \overline{x}_i$ and compose it with a homomorphism from q_T to $\mathcal{U}_{q_{i+1},\mathcal{O}}$ to obtain a contradiction to $(q_T, \mathcal{O})$-minimality of q_i.

Therefore, $\mathsf{var}(q_i) \subseteq \mathsf{img}(h_i^*)$ and $|\mathsf{var}(q_i)| \leq |\mathsf{var}(q_{i+1})|$ for all i. We thus focus on subsequences $q_\ell, \dots, q_k$ such that $|\mathsf{var}(q_\ell)| = \cdots = |\mathsf{var}(q_k)|$ in order to show that the length of the entire generalization sequence is bounded. Note that for all i with $\ell \leq i < k$, the mapping h_i^* is a bijection between $\mathsf{var}(q_{i+1})$ and $\mathsf{var}(q_i)$. With V_i we denote the set of all existential variables $x \in \mathsf{var}(q_i)$ which do not occur in a role atom in q_i, and define $U_i = \mathsf{var}(q_i) \setminus V_i$. Let us further denote with q^x the restriction $q|_{\{x\}}$ of a query q to a single variable $x \in \mathsf{var}(q)$. Using these sets, q_i can be written as

$$q_i(\overline{x}_i) \leftarrow q_i|_{U_i} \wedge \bigwedge_{x \in V_i} q_i^x.$$

By definition, each q_i^x, $x \in V_i$ is a query without answer variables, as all answer variables are in U_i.

Claim 1. $x \in U_{i+1}$ implies $h_i^*(x) \in U_i$.

Proof of Claim 1. Let $x \in U_{i+1}$. If x is an answer variable, then $h_i(x) = h_i^*(x)$ is an answer variable and thus in U_i. Suppose now that there is a role atom $R(x, y)$ in q_{i+1} and consider the pair $(h_i(x), h_i(y)) \in R^{\mathcal{U}_{q_i,\mathcal{O}}}$ which exists since h_i is a homomorphism. Since h_i^* is a bijection, either $y = x$ or $h_i^*(y) \neq h_i^*(x)$.

- In the first case, we obtain $R(z, z) \in \mathcal{U}_{q_i,\mathcal{O}}$ for $z = h_i(x) = h_i(y)$. The definition of $\mathcal{U}_{q_i,\mathcal{O}}$ yields $z \in \mathsf{var}(q_i)$, $R(z, z)$ occurs in q_i, and $h_i^*(x) = h_i(x) \in U_i$.
- In the second case, we obtain $h_i^*(x) = h_i(x) \in \mathsf{var}(q_i)$, $h_i^*(y) = h_i(y) \in \mathsf{var}(q_i)$ using the definition of h_i^*, and $R(h_i(x), h_i(y))$ occurs in q_i. Hence, $h_i^*(x) \in U_i$.

This completes the proof of Claim 1.

Claim 1 together with h_i^* being a bijection implies $|U_{i+1}| \leq |U_i|$ for every $i \in \{\ell, \dots, k\}$. We now consider subsequences $q_m, \dots, q_n$ of $q_\ell, \dots, q_k$ with $|U_m| = \cdots = |U_n|$ and thus $|V_m| = \cdots = |V_n|$. Since $|U_i| \leq |\mathsf{var}(q_T)|$, for all i, it suffices to show that the length of such a sequence is bounded by a polynomial in $|\mathsf{var}(q_T)|$ and $|\mathsf{sig}(\mathcal{O})|$.

Claim 2. For every $i \in \{m, \dots, n-1\}$,

1. h_i^* is a bijection between V_{i+1} and V_i, and
2. h_i is a bijection between U_{i+1} and U_i.

Proof of Claim 2. Point 1 is a consequence of Claim 1 as well as the facts that $|V_i| = |V_{i+1}|$ and h_i^* is a bijection between $\mathsf{var}(q_{i+1})$ and $\mathsf{var}(q_i)$.

For showing Point 2, assume for contradiction that there is some $x \in U_i$ such that for every $y \in U_{i+1}$, $x \neq h_i(y)$. Then, since h_i^* is a bijection, there must be some $y \in \mathsf{var}(q_{i+1})$ such that $h_i^*(y) = x$ and $h_i(y)$ is strictly within the subtree rooted at x in $\mathcal{U}_{q_i,\mathcal{O}}$, and this y is unique. We consider the following cases.

If $y \in V_{i+1}$, then $x \in V_i$ by the first point, a contradiction. If y is an answer variable, then $h_i(y)$ is an answer variable, not a proper trace within the subtree rooted at x, a contradiction. If $y \in U_{i+1}$ and y is not an answer variable, there must be an atom $R(y, z) \in q_{i+1}$. Since $h_i(y)$ is *strictly* below x, this leads to a contradiction as follows. If $z \neq y$, then $h_i^*(z) = h_i^*(y)$ contradicts the fact that h_i^* is a bijection. If, on the other hand, $z = y$, then h_i is not a homomorphism since there is no self-loop $R(h_i(y), h_i(y))$ in $\mathcal{U}_{q_i,\mathcal{O}}$, by definition of universal models. This completes the proof of Claim 2.

Sanctioned by the first point in Claim 2, in what follows we assume for the sake of readability that $h_i^*(x) = x$ for all $x \in V_{i+1}$ and $i \in \{m, \dots, n\}$. Hence, $V_m = \cdots = V_n$. Now, observe that since $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$, for all $i \in \{m, \dots, n-1\}$ one of the following must be the case:

1. the inverse of h_i is not a homomorphism from $\mathcal{U}_{q_i,\mathcal{O}}|_{U_i}$ to $\mathcal{U}_{q_{i+1},\mathcal{O}}|_{U_{i+1}}$;
2. there is some $x \in V_{i+1}$ such that $q_{i+1}^x \not\subseteq_{\mathcal{O}} q_i^x$.

Indeed, if neither Point 1 nor Point 2 is satisfied then $q_i \equiv_{\mathcal{O}} q_{i+1}$, in contradiction to the definition of generalization sequences. It thus remains to bound the number of times each of these points can be satisfied along $q_m, \dots, q_n$. For a finite interpretation $\mathcal{I}$ we mean the *number of occurrences of concept and role names* to refer to

$$\sum_{A \in \mathsf{N_C}} |A^{\mathcal{I}}| + \sum_{r \in \mathsf{N_R}} |r^{\mathcal{I}}|.$$

The following two claims bound Point 1 and Point 2, respectively.

Claim 3. The number of $i \in \{m, \dots, n\}$, such that the inverse of h_i is not a homomorphism from $\mathcal{U}_{q_i,\mathcal{O}}|_{U_i}$ to $\mathcal{U}_{q_{i+1},\mathcal{O}}|_{U_{i+1}}$, is at most $2 \cdot |\mathsf{var}(q_T)|^2 \cdot (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)$.

Proof of Claim 3. Let i be as in the claim. By Point 2 of Claim 2, h_i is a bijective homomorphism from $\mathcal{U}_{q_{i+1},\mathcal{O}}|_{U_{i+1}}$ to $\mathcal{U}_{q_i,\mathcal{O}}|_{U_i}$. Hence, the number n_{i+1} of occurrences

of concept and role names in $\mathcal{U}_{q_{i+1},\mathcal{O}}|_{U_{i+1}}$ is at most the number n_i of occurrences of concept and role names in $\mathcal{U}_{q_i,\mathcal{O}}|_{U_i}$. As the inverse of h_i is not a homomorphism, we have $n_{i+1} < n_i$. Since the number of occurrences of concept and role names in $\mathcal{U}_{q_i,\mathcal{O}}$ is bounded by

$$(|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|) \cdot |\mathsf{var}(q_T)|^2 + (|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|) \cdot |\mathsf{var}(q_T)|,$$

the claim follows. This completes the proof of Claim 3.

Claim 4. Let $x \in V_m$. The number of $i \in \{m, \dots, n-1\}$ such that $q^x_{i+1} \not\subseteq_{\mathcal{O}} q^x_i$ is at most $(|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|)^2$.

Proof of Claim 4. Let $i \in \{m, \dots, n-1\}$ with $q^x_{i+1} \not\subseteq_{\mathcal{O}} q^x_i$. We distinguish two cases.

(A) $h_i(x) = x$.

Since h_i is a homomorphism, the number n_{i+1} of occurrences of concept and role names $\mathcal{U}^x_{q_{i+1},\mathcal{O}}$ is at most the number n_i of occurrences of concept and role names in $\mathcal{U}^x_{q_i,\mathcal{O}}$. From $q^x_{i+1} \not\subseteq_{\mathcal{O}} q^x_i$, it follows that $n_{i+1} < n_i$.

(B) $h_i(x) \neq x$, that is, $h_i(x)$ is strictly within the subtree below x.

By definition of the universal model and since $\mathcal{O}$ is an $\mathcal{EL}^r$ ontology in normal form, there is an atom $A(x)$ in q^x_i such that there is a homomorphism from q^x_{i+1} to $\mathcal{U}_{\{A(a)\},\mathcal{O}}$. We claim that $A(x)$ is not an atom in any query q^x_j with $i < j \leq n-1$. Indeed, if $A(x)$ occurs in q^x_j for such j, then $q^x_j \subseteq_{\mathcal{O}} q^x_{i+1}$. The homomorphisms $h_i, \dots, h_j$ witness that $q^x_i \subseteq_{\mathcal{O}} q^x_{i+1} \subseteq_{\mathcal{O}} \cdots \subseteq_{\mathcal{O}} q^x_j$, and thus all these queries are actually equivalent, in contradiction to the choice of i.

Observe that Point (A) can only happen $|\mathsf{sig}(\mathcal{O})| + |\mathsf{sig}(q_1)|$ times, without Point (B) happening in between. Moreover, Point (B) can only happen $|\mathsf{sig}(\mathcal{O})|$ times overall. This completes the proof of Claim 4.

Since $|V_m| \leq |\mathsf{var}(q_T)|$, we obtain from Claims 3 and 4 that the length of the sequence $q_m, \dots, q_n$ is bounded by a polynomial in $|\mathsf{var}(q_T)|$, $|\mathsf{sig}(\mathcal{O})|$, and $|\mathsf{sig}(q_1)|$. □

As seen in Example 4.36 and the proof of Theorem 5.32, Boolean components of queries that are mapped into proper traces of the universal model have the potential to cause long generalization sequences. Fortunately, the universal model of $\mathcal{EL}^r$ ontologies is simple enough such that even sequences of CQs that are not rooted must approach q_T after a polynomial number of steps.

Extracting Chordal and Symmetry-free CQs

It remains to describe how a new hypothesis can be extracted from $C_{q_H,\mathcal{O}} \times C_{\mathcal{A},\mathcal{O}}$, that belongs to the desired query class. In Section 4.5, we defined the subroutine $\mathsf{extract}_{\mathrm{ELIQ}}$ to obtain a new ELIQ hypothesis from an arbitrary CQ q with $q \subseteq_{\mathcal{O}} q_T$ using membership queries. For learning $\mathrm{ELIQ}^{\mathrm{sf}}$ or $\mathrm{CQ}^{\mathrm{csf}}$ queries, $\mathsf{extract}_{\mathrm{ELIQ}}$ does not suffice, as the result of $\mathsf{extract}_{\mathrm{ELIQ}}$ may contain non-safe symmetries, and chordal cycles are removed. We thus need a new subroutine that obtains a hypothesis from $\mathrm{ELIQ}^{\mathrm{sf}}$ or $\mathrm{CQ}^{\mathrm{csf}}$ by properly handling symmetries, chordal cycles and queries of higher arity.

As we will see, handling non-unary CQs is not trivial. The subroutine that we define in this section will only handle CQs of fixed (but arbitrary) arity. We write $\mathrm{CQ}_w^{\mathrm{csf}}$ with $w \geq 0$ to refer to the class of $\mathrm{CQ}^{\mathrm{csf}}$ queries with exactly w answer variables. For every class $\mathcal{Q} \in \{\mathrm{ELQ}, \mathrm{ELIQ}^{\mathrm{sf}}\} \cup \{\mathrm{CQ}_w^{\mathrm{csf}} \mid w \geq 0\}$, we define a subroutine $\mathsf{extract}_{\mathcal{Q}}$. It takes as input the ontology $\mathcal{O}$ and a CQ $q(\overline{x})$ of the right arity such that $q \subseteq_{\mathcal{O}} q_T$ and produces a query $p(\overline{y}) \in \mathcal{Q}$ such that $q \subseteq_{\mathcal{O}} p \subseteq_{\mathcal{O}} q_T$ and p is $(q_T, \mathcal{O})$-minimal. To describe the workings of $\mathsf{extract}_{\mathcal{Q}}$, we define the notion of a *forbidden cycle*, depending on the query class $\mathcal{Q}$. If $\mathcal{Q} \in \{\mathrm{ELQ}, \mathrm{ELIQ}^{\mathrm{sf}}\}$, then every cycle is forbidden. If $\mathcal{Q} = \mathrm{CQ}_w^{\mathrm{csf}}$ for some $w \geq 0$, then every chordless cycle of length at least four that contains at least one existential variable is forbidden.

The subroutine $\mathsf{extract}_{\mathcal{Q}}$ starts by setting $p(\overline{y}) = \mathsf{minimize}_{\mathcal{O}}(q)$ and then exhaustively applies the following operations.

Expand cycle. Choose a forbidden cycle $R_1(x_1, x_2), \dots, R_n(x_n, x_1)$ in p, and introduce fresh variables $x_1', \dots, x_n'$. Then

1. remove all atoms of the form $R(x_n, x_1)$,
2. add the atom $A(x_i')$ for all $A(x_i) \in p$ and $1 \leq i \leq n$,
3. add $R(x_i', y)$ for all $R(x_i, y) \in p$ with $1 \leq i \leq n$ and $y \in \mathsf{var}(p) \setminus \{x_1, \dots, x_n\}$,
4. add $R(x_i', x_j')$ for all $R(x_i, x_j) \in p$ with $1 \leq i, j \leq n$ and $\{i, j\} \neq \{1, n\}$,
5. add $R(x_n, x_1')$ and $R(x_n', x_1)$ for all $R(x_n, x_1) \in p$.

Let Y be the set of tuples obtained from $\overline{y} = (y_1, \dots, y_w)$ by replacing any number of components y_j with y_j'. Use membership queries to identify a $\overline{y}' \in Y$ such that $\mathcal{A}_p, \mathcal{O} \models q_T(\overline{y}')$. Use this tuple $\overline{y}'$ as the new answer variables of p. Apply $\mathsf{minimize}_{\mathcal{O}}$ to the result.

Split symmetry. Choose a symmetry $r(y_1, x), r(y_2, x) \in p(\overline{y})$ that is not safe. Introduce a fresh variable x', add atoms $A(x')$ for all $A(x) \in p$ and $S(y, x')$ for all atoms $S(y, x) \in p$ with $S(y, x) \neq r(y_1, x)$. Remove the atom $r(y_2, x)$. Apply $\mathsf{minimize}_{\mathcal{O}}$ to the result.

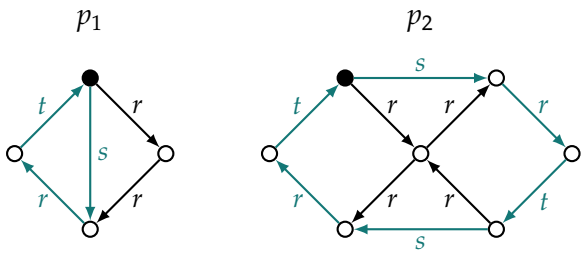

Figure 5.7: An application of *Expand cycle*. The chosen cycle is marked in green.

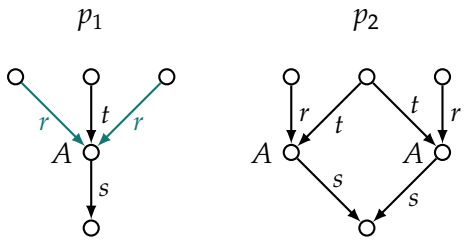

Figure 5.8: An application of *Split symmetry*. The chosen symmetry is marked in green.

Note that the running time of *Expand cycle* and the number of membership queries it performs depend exponentially on the arity of q_T. *Expand cycle* is not the same operation as *Double cycle* from Section 4.5, as *Expand cycle* needs to be more careful to preserve matches from CQ^{csf} queries and handle non-unary queries. We refer to the fresh variables $x'_1, \dots, x'_n$ introduced by *Expand cycle* as *copies* of $x_1, \dots, x_n$.

Example 5.33. Consider the CQ p_1 displayed in Figure 5.7 and assume $\mathcal{Q} = \text{ELIQ}^{\text{sf}}$. It contains the cycle $s(x_1, x_2), r(x_2, x_3), t(x_3, x_1)$, which is forbidden for ELIQ^{sf} queries. Applying *Expand cycle* to this cycle results in the CQ p_2 displayed in Figure 5.7, to which $\mathsf{minimize}_{\mathcal{O}}$ is then applied. Compared to the operation *Double cycle* in Example 4.38, variables that are not on the cycle are not duplicated. This makes some roles that were functional in p_1 no longer functional in p_2.

The exhaustive application of the second operation *Split symmetry* eliminates all non-safe symmetries, and thus produces a symmetry-free query.

Example 5.34. The Boolean CQ p_1 displayed in Figure 5.8 contains a non-safe symmetry $r(y_1, x), r(y_2, x)$. To eliminate it, *Split symmetry* first produces the CQ p_2 displayed in Figure 5.8 and then applies $\mathsf{minimize}_{\mathcal{O}}$. Note that p_2 contains a new non-safe symmetry, namely $s(x, y_3), s(x', y_3)$. We later show that although new symmetries can be created, *Split symmetry* can only be applied a polynomial number of times.

Before we show that $\mathsf{extract}_{\mathcal{Q}}$ runs in polynomial time and that it indeed produces a query from $\mathcal{Q}$, we first analyze the two operations that $\mathsf{extract}_{\mathcal{Q}}$ applies. We begin with *Expand cycle*.

Lemma 5.35. *Let p be a CQ and p' the result of applying* Expand cycle *to p, but before minimization. Then,*

1. *$p', x \preceq_{\mathcal{EL}} p, x$ and $p, x \preceq_{\mathcal{EL}} p', x$ for all $x \in \mathsf{var}(p)$, and*
2. *$p', x' \preceq_{\mathcal{EL}} p, x$ and $p, x \preceq_{\mathcal{EL}} p', x'$ for all variables x' that are copies of x.*

Proof. This can be shown using the simulation

$$S = \{(x, x) \in \mathsf{var}(p)^2\} \cup \{(x, x') \in \mathsf{var}(p) \times \mathsf{var}(p') \mid x' \text{ is a copy of } x\}$$

from p to p' and its inverse S^- which is a simulation from p' to p. □

Lemma 5.36. *Let $q_T \in \mathcal{Q}$, $q(\overline{x})$ be a $(q_T, \mathcal{O})$-minimal CQ and $p(\overline{y})$ the result of applying* Expand cycle *to q, but before minimization. Then*

1. *$q \subseteq_{\mathcal{O}} p$,*
2. *$p \not\subseteq_{\mathcal{O}} q$, and*
3. *if $q \subseteq_{\mathcal{O}} q_T$, then $p \subseteq_{\mathcal{O}} q_T$.*

Proof. Proving the first two points is very similar to the proof of Lemma 4.41.

For showing Point 1, we define the natural mapping h from $\mathsf{var}(p)$ to $\mathsf{var}(q)$ by setting $h(x) = x$ for all original variables x and $h(x') = x$ for all newly introduced copies x' of variables x. By construction of p, h is a homomorphism from p to q with $h(\overline{y}) = \overline{x}$. It is also a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$ and therefore $q \subseteq_{\mathcal{O}} p$.

In order to show Point 2, assume to the contrary that $p \subseteq_{\mathcal{O}} q$. Then, there is a homomorphism g from q to $\mathcal{U}_{p,\mathcal{O}}$ with $g(\overline{x}) = \overline{y}$. Composing g with the extension h^+ of h to a homomorphism from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$, which exists by Lemma 3.8, yields a homomorphism $\widehat{g}$ from q to $\mathcal{U}_{q,\mathcal{O}}$ with $\widehat{g}(\overline{x}) = \overline{x}$. From Lemma 5.31 and $(q, \mathcal{O})$-minimality of q it follows that $\widehat{g}^*$ must be injective, which implies that $\widehat{g}$ must be injective.

Let $R_1(y_1, y_2), \dots, R_n(y_n, y_1)$ be the cycle that was expanded in the construction of p and consider the set Γ of all sets of variables that form a cycle of length n in $\mathcal{U}_{p_i,\mathcal{O}}$. For example, $\{y_1, \dots, y_n\} \in \Gamma$.

Let $\{x_1, \dots, x_n\}$ be any element of Γ. We show that $\{\widehat{g}(x_1), \dots, \widehat{g}(x_n)\} \in \Gamma$. If for some x_i, $\widehat{g}(x_i)$ is a proper trace in $\mathcal{U}_{q,\mathcal{O}}$, then $\widehat{g}^*$ is not injective, a contradiction. Since $\widehat{g}$ is a homomorphism, there is an atom $R(\widehat{g}(x_i), \widehat{g}(x_{i+1})) \in q$ if there is an atom

$R(x_i, x_{i+1}) \in q$. Thus, in order to show that $\{\widehat{g}(x_1), \dots, \widehat{g}(x_n)\} \in \Gamma$, it suffices to show that $\widehat{g}(x_1), \dots, \widehat{g}(x_n)$ are all pairwise different. Assume the contrary. Then there are x_j and x_k with $x_j \neq x_k$ and $\widehat{g}(x_j) = \widehat{g}(x_k)$, implying that $\widehat{g}$ is not injective. This in turn implies that $\widehat{g}^*$ is not injective, a contradiction.

Hence, we can define a function $f : \Gamma \to \Gamma$ by setting

$$f(\{x_1, \dots, x_n\}) = \{\widehat{g}(x_1), \dots, \widehat{g}(x_n)\}$$

for all $\{x_1, \dots, x_n\} \in \Gamma$. Assume that there are sets $\gamma, \gamma' \in \Gamma$ with $\gamma \neq \gamma'$ and $f(\gamma) = f(\gamma')$. Since $\gamma \neq \gamma'$ and $|\gamma| = |\gamma'|$, there must be a variable $x \in \gamma$ with $x \notin \gamma'$. Since $f(\gamma) = f(\gamma')$, there is a variable $x' \in \gamma'$ with $\widehat{g}(x) = \widehat{g}(x')$, and clearly $x' \neq x$, a contradiction. Therefore, f is a bijection from Γ to Γ.

Since Γ is finite, it follows that there must be a $j \geq 1$ such that $f^j(\{y_1, \dots, y_n\}) = \{y_1, \dots, y_n\}$. By definition of f, this implies that $\{\widehat{g}^j(y_1), \dots, \widehat{g}^j(y_n)\} = \{y_1, \dots, y_n\}$. Recall that $\widehat{g}$ is the composition of the homomorphism g from q to $\mathcal{U}_{p,\mathcal{O}}$ and the homomorphism h^+ from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$. Since $(q_T, \mathcal{O})$-minimality of q implies that $\widehat{g}$ is injective by Lemma 4.21, g must also be injective. Thus, composing $\widehat{g}^{j-1}$ and g yields an injective homomorphism g' that maps the cycle $\{y_1, \dots, y_n\}$ in q to some subset of the expanded cycle $\{y_1, y_1', \dots, y_n, y_n'\}$ in $\mathcal{U}_{p,\mathcal{O}}$. We distinguish cases.

First, consider the case where $\{g'(y_1), \dots, g'(y_n)\} = \{y_1, \dots, y_n\}$. By the construction of p from q, the restriction of $\mathcal{U}_{p_i',\mathcal{O}}$ to $\{y_1, \dots, y_n\}$ contains one less role than the restriction of $\mathcal{U}_{p_i,\mathcal{O}}$ to $\{y_1, \dots, y_n\}$, implying that g' cannot be an injective homomorphism, leading to a contradiction. The case where $\{g'(y_1), \dots, g'(y_n)\} = \{y_1', \dots, y_n'\}$ is analogous.

The remaining case is that $\{g'(y_1), \dots, g'(y_n)\}$ contains both variables of the form y_j and y_j'. Then, there must be two different atoms in the cycle $R_1(y_1, y_2), \dots, R_n(y_n, y_1)$ that are mapped by g' to the role atoms $r(x, y'), r(x', y)$ that were added by *Expand cycle* to connect the disjoint copy of q. However, since $h(x') = h(x)$ and $h(y') = h(y)$, this implies that the composition of g' and h^+ is a non-injective homomorphism from q to $\mathcal{U}_{q,\mathcal{O}}$, again contradicting $(q_T, \mathcal{O})$-minimality of q.

To show Point 3, let $R_1(x_1, x_2), \dots, R_n(x_n, x_1) \in q$ be the cycle that was expanded during construction of p from $q(\overline{x})$, and let h be a homomorphism from $q_T(\overline{x}_0)$ to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{x}_0) = \overline{x}$. We construct a homomorphism g from q_T to $\mathcal{U}_{p,\mathcal{O}}$ with $g(\overline{x}_0) = \overline{x}'$ for some $\overline{x}' \in Y$. For this, we partition $\mathsf{var}(q_T)$ into sets M_0, M_1, M_2 such that:

- $x \in M_0$ if $h(x) \in \{x_1, \dots, x_n\}$, that is, $h(x)$ lies on the expanded cycle;
- $x \in M_1$ if $h(x) \notin \mathsf{var}(q)$, that is, $h(x)$ is a proper trace in $\mathcal{U}_{q,\mathcal{O}}$ generated by existential quantification;
- all other variables are in M_2.

We start by setting

$$g(x) = h(x) \quad \text{for all } x \in M_2.$$

To define $g(x)$ for the variables in $x \in M_0$, we first construct an auxiliary query q'_T, that takes the form of a disjoint union of not necessarily directed trees with multi-edges and self-loops[1]. If $\mathcal{Q} \in \{\text{ELQ}, \text{ELIQ}^{\text{sf}}\}$, then q'_T is simply q_T. If $\mathcal{Q} = \text{CQ}^{\text{csf}}_w$, then q'_T is obtained by starting with the restriction of q_T to the variables in M_0 and then exhaustively choosing and identifying variables x, x' such that

1. there is a cycle $R_0(y_0, y_1), R_1(y_1, y_2), R_3(y_2, y_0)$ with $\{x, x'\} \subseteq \{y_0, y_1, y_2\} \subseteq M_0$ and,

2. $h(x) = h(x')$.

Note that this process may also identify answer variables. The result of identifying an answer variable and an existential variable is an answer variable.

Next, we observe that since q_T is chordal, all CQs $q_T = p_1, \dots, p_k = q'_T$ encountered during the construction of q'_T are chordal as well. We show this by induction on the index i of the CQ p_i. In the induction start, it follows directly that p_1 is chordal, since q_T is chordal. In the induction step, assume that p_i contains a forbidden cycle $C = S_1(z_1, z_2), \dots, S_n(z_n, z_1)$ of length at least four containing at least one existential variable. Then p_{i-1} contains C or a cycle C' that can be obtained from C by replacing some edge $S_j(z_j, z_{j+1})$ with two edges $S_{j,1}(z_j, u), S_{j,2}(u, z_{j+1})$ because u and z_{j+1} were identified when constructing p_i. In the first case, C has a chord in p_{i-1} and thus also in p_i. In the second case, C' contains at least one existential variable since C does and consequently has a chord in p_{i-1}. If this chord is not between z_j and z_{j+1}, then C contains a chord in p. If the chord is between z_j and z_{j+1}, then we are in the first case.

We now show that q'_T takes the form of a disjoint union of not necessarily directed trees with multi-edges and self loops. Assume to the contrary that q'_T contains a cycle C of length exceeding 2. If there is an existential variable x on C, then q'_T being chordal implies that x occurs on a cycle of length 3, in contradiction to the construction of q'_T. Now assume that there is no existential variable on C. As the image of C under h is a cycle in $\mathcal{U}_{q,\mathcal{O}}$ and the cycle chosen by the *Expand cycle* step is chordless, the image of C under h must contain *all* variables $\{x_1, \dots, x_n\}$. Since all variables on C are answer variables, this means that all variables in M_0 are from $\overline{x}$, in contradiction to the fact that $\{x_1, \dots, x_n\}$ contains at least one existential variable.

This finishes the construction of q'_T. For defining $g(x)$ for the variables $x \in \mathsf{var}(q'_T)$, we start at some arbitrary variable in each tree in q'_T and then follow the tree

[1] Or alternatively: has treewidth 1.

structure, switching between the variables $x_1, \dots, x_n$ and their copies $x'_1, \dots, x'_n$ as necessary. Next, we make this precise.

For each connected component of q'_T, choose an arbitrary variable z from that component and set $g(z) = h(z)$. Then, exhaustively apply the following rule: if q'_T contains an atom $R(x, y)$ with $g(y)$ defined and $g(x)$ undefined, set

- $g(x) = h(x)$ if $g(y) = x_i$ and either $h(x) = x_{i+1}$ and $i < n$ or $h(x) = x_{i-1}$ and $i > 0$;
- $g(x) = h(x)'$ if $g(y) = x'_i$ and either $h(x) = x_{i+1}$ and $i < n$ or $h(x) = x_{i-1}$ and $i > 0$;
- $g(x) = x'_1$ if $g(y) = x_n$ and $h(x) = x_1$;
- $g(x) = x_1$ if $g(y) = x'_n$ and $h(x) = x_1$;
- $g(x) = x'_n$ if $g(y) = x_1$ and $h(x) = x_n$;
- $g(x) = x_n$ if $g(y) = x'_1$ and $h(x) = x_n$.

It can be verified that by construction of p in all cases $R(g(x), g(y)) \in p$. Next, we extend g to all variables in M_0 by setting $g(y) = g(x)$ if y was identified with $x \in \mathsf{var}(q'_T)$ during the construction of q'_T (note that this implies $h(y) = h(x)$).

It remains to define $g(x)$ for the variables $x \in M_1$. By definition of M_1, $h(x)$ is a trace yw with $y \in \mathsf{var}(q)$ and $w \neq \varepsilon$, that is, x is mapped in $\mathcal{U}_{q,\mathcal{O}}$ to a proper trace that starts with y. Now do the following:

- if there is a path in q_T from some variable $z \in M_0$ to x, then choose a z such that the path is shortest (thus, $h(z) = y$ and $g(z)$ has already been defined) and set $g(x) = g(z)w$;
- otherwise, set $g(x) = h(x)$.

This is well-defined, since

1. for each $y \in \mathsf{var}(q)$, the subtrees below y in $\mathcal{U}_{q,\mathcal{O}}$ and in $\mathcal{U}_{p,\mathcal{O}}$ are identical,
2. for $1 \leq i \leq n$, the subtree below x_i in $\mathcal{U}_{q,\mathcal{O}}$ and the subtree below x'_i in $\mathcal{U}_{p,\mathcal{O}}$ are identical,

due to Lemma 5.35.

This completes the definition of g. By definition of g, it follows that $g(x) \in \{h(x), h(x)'\}$ for all $x \in M_0$ and $g(\overline{x}_0) \in Y$ as announced. Set $\overline{y} = g(\overline{x}_0)$. To prove that $p(\overline{y}) \subseteq_{\mathcal{O}} q_T(\overline{x}_0)$ it remains to show that g is a homomorphism from q_T to $\mathcal{U}_{p,\mathcal{O}}$.

First, let $A(x)$ be a concept atom in q_T. Then $h(x) \in A^{\mathcal{U}_{q,\mathcal{O}}}$. By Lemma 5.35 and the definition of $\mathcal{U}_{p,\mathcal{O}}$, it then follows that $g(x) \in A^{\mathcal{U}_{p,\mathcal{O}}}$.

Now let $R(z_1, z_2)$ be a role atom in q_T. We distinguish cases according to z_1, z_2 belonging to M_0, M_1, M_2:

- If $z_1, z_2 \in M_0$, then q'_T contains an atom $R(z'_1, z'_2)$ such that each z_i was identified with z'_i during the construction of q'_T. If $z'_1 \neq z'_2$, then $R(g(z'_1), g(z'_2)) \in p$, as argued in the definition of g for variables from q'_T. If $z'_1 = z'_2$, the same is true due to the construction of p. In both cases, $g(z_i) = g(z'_i)$ for $i \in \{1, 2\}$. Thus, $(g(z_1), g(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$, as required.

- If $z_1, z_2 \in M_1$, then $h(z_1) = yv$ and $h(z_2) = yw$ for some $y \in \mathsf{var}(q)$ and some non-empty sequences v, w, and $(h(z_1), h(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$. By definition of g, we have $g(z_1) = \widehat{y}v$ and $g(z_2) = \widehat{y}w$ for some $\widehat{y} \in \{y, y'\}$. By Lemma 5.35, the subtree below $\widehat{y}$ in $\mathcal{U}_{p,\mathcal{O}}$ is identical to the subtree below y in $\mathcal{U}_{q,\mathcal{O}}$. This implies $(g(z_1), g(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$.

- If $z_1, z_2 \in M_2$, then $g(z_1) = h(z_1)$, $g(z_2) = h(z_2)$, and $R(h(z_1), h(z_2)) \in q$ because h is a homomorphism from q_T to $\mathcal{U}_{q,\mathcal{O}}$ and $h(z_1), h(z_2) \in \mathsf{var}(q)$. Since, additionally, $h(z_1), h(z_2) \notin \{x_1, \dots, x_n\}$, $R(h(z_1), h(z_2)) \in p$ and thus $(g(z_1), g(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$.

- If $z_1 \in M_0$ and $z_2 \in M_1$, then $h(z_1) \in \{x_1, \dots, x_n\}$ and $h(z_2)$ takes the form $h(z_1)rM$. Moreover, $g(z_1) \in \{h(z_1), h(z_1)'\}$ and $g(z_2) = g(z_1)rM$. It thus follows from Lemma 5.35 that $(h(z_1), h(z_2)) \in R^{\mathcal{U}_{q,\mathcal{O}}}$ and from the construction of universal models that $(g(z_1), g(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$.

- If $z_1 \in M_0$ and $z_2 \in M_2$, then $h(z_1) \in \{x_1, \dots, x_n\}$ and $h(z_2) \in \mathsf{var}(q) \setminus \{x_1, \dots, x_n\}$. Moreover, $g(z_1) \in \{h(z_1), h(z_1)'\}$ and $g(z_2) = h(z_2)$. It follows from $(h(z_1), h(z_2)) \in R^{\mathcal{U}_{q,\mathcal{O}}}$ that $R(h(z_1), h(z_2)) \in q$. By construction of p, we thus have $R(g(z_1), g(z_2)) \in p$ and $(g(z_1), g(z_2)) \in R^{\mathcal{U}_{p,\mathcal{O}}}$.

- If $z_1 \in M_1$, $z_2 \in M_2$, then $h(z_2) \in \mathsf{var}(q) \setminus \{x_1, \dots, x_n\}$, $h(z_1)$ takes the form $h(z_2)rM$ and $R = r^-$. Moreover, $g(z_i) = h(z_i)$ for $i \in \{1, 2\}$ and it remains to use Lemma 5.35 as in previous cases. □

We continue with the properties of *Split symmetry*.

Lemma 5.37. *Let $q(\overline{x})$ be a $(q_T, \mathcal{O})$-minimal CQ and $p(\overline{y})$ the result of applying* Split symmetry *to q, but before minimization. Then*

1. *$q \subseteq_{\mathcal{O}} p$,*
2. *$p \not\subseteq_{\mathcal{O}} q$, and*
3. *if $q \subseteq_{\mathcal{O}} q_T$, then $p \subseteq_{\mathcal{O}} q_T$.*

Proof. Let $r(y_1, x), r(y_2, x)$ be the non-safe symmetry that is split during the application of *Split symmetry*. In order to show Point 1, we define a homomorphism h from

p to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{y}) = \overline{x}$. Set $h(z) = z$ for all $z \in \mathsf{var}(q)$ and $h(x') = x$. By construction of p, h is a homomorphism, as required.

To show Point 2, assume for contradiction that $p \subseteq_{\mathcal{O}} q$ and let g be a homomorphism from q to $\mathcal{U}_{p,\mathcal{O}}$ with $g(\overline{x}) = \overline{y}$. Let h^+ be the extension of the homomorphism h to a homomorphism from $\mathcal{U}_{p,\mathcal{O}}$ to $\mathcal{U}_{q,\mathcal{O}}$ with $h^+(\overline{y}) = \overline{x}$, which exists by Lemma 3.8. The composition $\widehat{g}$ of g and h^+ is then a homomorphism from q to $\mathcal{U}_{q,\mathcal{O}}$ with $\widehat{g}(\overline{x}) = \overline{x}$. By Lemma 5.31, $\widehat{g}^*$ must be injective, which implies that $\widehat{g}$ is injective. Therefore, every symmetry $r'(z_1, z), r'(z_2, z)$ in q must be mapped by $\widehat{g}$ to some symmetry in q, such that no two different symmetries are mapped to the same symmetry. Finiteness of q then implies that there must be a symmetry $r(z_1, z), r(z_2, z) \in q$ with $\widehat{g}(z) = x$, $\widehat{g}(z_1) = y_1$ and $\widehat{g}(z_2) = y_2$. But this implies $g(z) = x \in \mathsf{var}(p)$ or $g(z) = x' \in \mathsf{var}(p)$. In the first case, $(g(z_2), g(z)) \notin r^{\mathcal{U}_{p,\mathcal{O}}}$ by construction of p, contradicting that g is a homomorphism. In the second case, $(g(z_1), g(z)) \notin r^{\mathcal{U}_{p,\mathcal{O}}}$ by construction of p, again contradicting that g is a homomorphism. Hence, $p \not\subseteq_{\mathcal{O}} q$.

For Point 3, assume for contradiction that $q \subseteq_{\mathcal{O}} q_T$ and $p \not\subseteq_{\mathcal{O}} q_T$. Let h be a homomorphism from q_T to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{x}_0) = \overline{x}$. Since $p \not\subseteq_{\mathcal{O}} q_T$, there must be atoms $r(z_1, z), r(z_2, z) \in q_T$ with $h(z_1) = y_1$, $h(z_2) = y_2$ and $h(z) = x$. These atoms must form a safe symmetry, since q_T is symmetry-free. The atoms $r(y_1, x), r(y_2, x) \in q$, however, must form a non-safe symmetry, as *Split symmetry* chose this symmetry. Consider the reason for $r(z_1, z), r(z_2, z) \in q_T$ being a safe symmetry. If z is an answer variable, then x must also be an answer variable, meaning that $r(y_1, x), r(y_2, x)$ is safe, a contradiction. If $r(z_1, z)$ or $r(z_2, z)$ occur on a cycle, then, by chordality of q_T, the same atom also occurs on a cycle of length 3. Since h is a homomorphism, this implies that either $r(y_1, x)$ or $r(y_2, x)$ also occur on a cycle, or that there is an atom $s(y_1, y_1)$, $s(x, x)$ or $s(y_2, y_2)$. In all cases, $r(y_1, x), r(y_2, x)$ is safe, again contradicting non-safety. If there is an atom $s(z_1, z_1)$, $s(z, z)$, or $s(z_2, z_2)$, the same atom must exist in q since h is a homomorphism, again contradicting that $r(y_1, x)$, $r(y_2, x)$ is not safe.

If no atom of the symmetry occurs on a cycle and $\overline{x}_0$ consists of a single answer variable $\widehat{x}$, and additionally $\mathsf{dist}(\widehat{x}, z) \neq \infty$ and $\mathsf{dist}(\widehat{x}, z) + 1 = \mathsf{dist}(\widehat{x}, z_1) = \mathsf{dist}(\widehat{x}, z_2)$, then there is a simple path $R_1(x_1, x_2), \dots, R_n(x_n, x_{n+1}) \in q_T$ with $x_1 = \widehat{x}$ and $x_{n+1} = z$ and $R_n(x_n, x_{n+1}) \notin \{r(z_1, z), r(z_2, z)\}$. Let the length of this path be n. Since h is a homomorphism, the image of this path in q must also be a path (but not necessarily a simple path) of length $m < n$. If for $i \in \{1, 2\}$ there is a simple path of length $< m$ from $h(\widehat{x})$ to y_i in q, then the atom $r(y_i, x)$ occurs on a cycle in q, and the symmetry $r(y_1, x), r(y_2, x)$ is safe, a contradiction. If there is no such simple path, then $r(y_1, z), r(y_2, z)$ is safe since $\mathsf{dist}(h(\widehat{x}), x) + 1 = \mathsf{dist}(h(\widehat{x}), y_1) = \mathsf{dist}(h(\widehat{x}), y_2)$, a contradiction. □

Lemmas 5.36 and 5.37 tell us that the sequence of queries produced by $\mathsf{extract}_Q$ forms a generalization sequence towards q_T. It remains to apply Theorem 5.32 to

Algorithm 5.3: Learning algorithm for ELQs / ELIQ^{sf} / CQ_w^{csf} under $\mathcal{EL}^r$ ontologies

For $\mathcal{Q} \in \{\text{ELQ}, \text{ELIQ}^{\text{sf}}\} \cup \{\text{CQ}_w^{\text{csf}} \mid w \geq 0\}$ and $q_T \in \mathcal{Q}$.
Input A signature Σ and an $\mathcal{EL}^r$ ontology $\mathcal{O}$ in normal form.
Output A $q_H \in \mathcal{Q}$ such that $q_H \equiv_{\mathcal{O}} q_T$.

$q_H^0 :=$ initial-CQ$(\Sigma, \mathcal{O})$
$q_H :=$ extract$_{\mathcal{Q}}(\mathcal{O}, q_H^0)$
while the equivalence query "$q_H \equiv_{\mathcal{O}} q_T$?" returns a counterexample $(\mathcal{A}, \overline{a})$ **do**
  $q'_H(\overline{x} \otimes \overline{a}) := C^3_{q_H,\mathcal{O}} \times C^3_{\mathcal{A},\mathcal{O}}$
  $q_H :=$ extract$_{\mathcal{Q}}(\mathcal{O}, q'_H)$
end while
return q_H

obtain a bound on the number of applications of *Expand cycle* and *Split symmetry*.

Lemma 5.38. *Let $\mathcal{Q} \in \{\text{ELQ}, \text{ELIQ}^{\text{sf}}\} \cup \{\text{CQ}_w^{\text{csf}} \mid w \geq 0\}$, $\mathcal{O}$ be an $\mathcal{EL}^r$ ontology in normal form and q a CQ such that $q \subseteq_{\mathcal{O}} q_T$. Then, the subroutine* extract$_{\mathcal{Q}}(\mathcal{O}, q)$ *terminates in time polynomial in $\|\mathcal{O}\| + \|q\| + \|q_T\|$ and returns a query $p \in \mathcal{Q}$ that is $(q_T, \mathcal{O})$-minimal and satisfies $q \subseteq_{\mathcal{O}} p \subseteq_{\mathcal{O}} q_T$.*

Proof. Let $p_1, p_2, \ldots$ be the sequence of queries produced by applying the operations *Expand cycle* and *Split symmetry*. By Lemma 5.36 and Lemma 5.37, $p_1, p_2, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$. As the operations *Expand cycle* and *Split symmetry* both ensure $(q_T, \mathcal{O})$-minimality of their result, every p_i is $(q_T, \mathcal{O})$-minimal. Hence, Theorem 5.32 implies a bound on the length of this sequence that is polynomial in $\|\mathcal{O}\| + \|q\| + \|q_T\|$. Neither *Expand cycle* nor *Split symmetry* are applicable to the last query p_n of this sequence. Therefore, p_n cannot contain forbidden cycles and must be symmetry-free. □

The Learning Algorithm

We now have all the necessary ingredients to formulate a learning algorithm for CQ^{csf} queries under $\mathcal{EL}^r$ ontologies: a CQ^{csf}-universal model that we can use in products and a subroutine that extracts $(q_T, \mathcal{O})$-minimal queries from the right query class using membership queries. The full algorithm is displayed as Algorithm 5.3. Since $\mathcal{EL}^r$ ontologies do not contain disjointness or functionality constraints, the initial CQ can simply be

$$q_H^0(x_0, \ldots, x_0) \leftarrow \bigwedge_{A \in \Sigma \cap \mathsf{N_C}} A(x_0) \wedge \bigwedge_{r \in \Sigma \cap \mathsf{N_R}} r(x_0, x_0).$$

Note that the restriction to ontologies in normal form is not essential as Lemma 4.8 holds for ELQs under $\mathcal{EL}^r$ ontologies, and can be extended to ELIQ^{sf} and CQ^{csf} queries.

We show that Algorithm 5.3 is a polynomial time learning algorithm. As in the similar proofs before, we show that the sequence $q_1, q_2, \ldots$ of assignments to q_H is a generalization sequence towards q_T under $\mathcal{O}$. Polynomial running time of Algorithm 5.3 then follows from the $(q_T, \mathcal{O})$-minimality of all q_i and Theorem 5.32, which yields the following main result of this section.

Theorem 5.39. *Let $\mathcal{Q} \in \{\text{ELQ}, \text{ELIQ}^{\text{sf}}\} \cup \{\text{CQ}_w^{\text{csf}} \mid w \geq 0\}$. $\mathcal{Q}$ queries are polynomial time learnable under $\mathcal{EL}^r$ ontologies using both equivalence and membership queries.*

Proof. Let $q_1, q_2, \ldots$ be the sequence of assignments to q_H during a run of Algorithm 5.3. We show that this sequence is a generalization sequence towards q_T under $\mathcal{O}$.

First, we show that $q_i \subseteq_{\mathcal{O}} q_T$ for all i. We show this by induction on i. Since $q_H^0 \subseteq_{\mathcal{O}} q_T$ and $q_1 = \text{extract}_{\mathcal{Q}}(q_H^0)$, this holds for $i = 1$ by Lemma 5.38. In the induction step, assume that $q_i \subseteq_{\mathcal{O}} q_T$. Then, by Lemma 5.28, $q_T(\overline{x}_0) \to \mathcal{C}^3_{q_i,\mathcal{O}}, \overline{x}_i$. Additionally, the counterexample $(\mathcal{A}, \overline{a})$ returned from the equivalence query must be such that $\mathcal{A}, \mathcal{O} \models q_T(\overline{a})$. Lemma 5.28 then implies that $q_T(\overline{x}_0) \to \mathcal{C}^3_{\mathcal{A},\mathcal{O}}, \overline{a}$. Hence, $q_T(\overline{x}_0) \to \mathcal{C}^3_{q_H,\mathcal{O}} \times \mathcal{C}^3_{\mathcal{A},\mathcal{O}}, \overline{x} \otimes \overline{a}$ by the properties of products. It follows that $q'_H \subseteq_{\mathcal{O}} q_T$ and therefore $q_{i+1} \subseteq_{\mathcal{O}} q_T$ by Lemma 5.38.

Next, we show that $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$ for all i. Recall that since $q_i \subseteq_{\mathcal{O}} q_T$, the counterexample $(\mathcal{A}, \overline{a})$ returned from the equivalence query must be such that $\mathcal{A}, \mathcal{O} \not\models q_i(\overline{a})$. Since $q'_H(\overline{x}') = \mathcal{C}^3_{q_i,\mathcal{O}} \times \mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ with $\overline{x}' = \overline{x} \otimes \overline{a}$, it follows that $q'_H(\overline{x}') \to \mathcal{C}^3_{q_i,\mathcal{O}}, \overline{x}_i$ and $q'_H(\overline{x}') \to \mathcal{C}^3_{\mathcal{A},\mathcal{O}}, \overline{a}$ by properties of products. Using the homomorphism from the universal model to the compact model, we also obtain that $\mathcal{U}_{q'_H,\mathcal{O}}, \overline{x}' \to \mathcal{C}^3_{q_i,\mathcal{O}}, \overline{x}_i$ and $\mathcal{U}_{q'_H,\mathcal{O}}, \overline{x}' \to \mathcal{C}^3_{\mathcal{A},\mathcal{O}}, \overline{a}$. Since $q'_H \subseteq_{\mathcal{O}} q_{i+1}$ by Lemma 5.38, it follows that $q_{i+1}(\overline{x}_{i+1}) \to \mathcal{C}^3_{q_i,\mathcal{O}}, \overline{x}_i$ and $q_{i+1}(\overline{x}_{i+1}) \to \mathcal{C}^3_{\mathcal{A},\mathcal{O}}, \overline{a}$ by Lemma 3.5 and composition of homomorphisms. As $q_{i+1} \in \mathcal{Q}$, it follows by Lemma 5.28 that $q_i \subseteq_{\mathcal{O}} q_{i+1}$ and $q_{\mathcal{A}} \subseteq_{\mathcal{O}} q_{i+1}$. The second query implication then implies that $q_{i+1} \not\subseteq_{\mathcal{O}} q_i$ since $q_{\mathcal{A}} \not\subseteq_{\mathcal{O}} q_i$.

Hence, $q_1, q_2, \ldots$ is a generalization sequence towards q_T under $\mathcal{O}$. Since all q_i are $(q_T, \mathcal{O})$-minimal by Lemma 5.38, we can apply Theorem 5.32 to show that Algorithm 5.3 must terminate after a polynomial number of iterations with a query $q_n \in \mathcal{Q}$ such that $q_n \equiv_{\mathcal{O}} q_T$. □

Note, though, that Algorithm 5.3, or more precisely, the subroutine $\text{extract}_{\mathcal{Q}}$, uses a number of membership queries that is exponential in the arity of the target CQ, and we only achieve polynomial time learnability by assuming that the arity is fixed. In the next section, we will discuss how we can improve upon this result.

5.4 Handling Queries of Unbounded Arity

In Section 5.3, we have seen that CQ^{csf} queries of *fixed arity* are learnable in polynomial time using membership queries and equivalence queries. In this section, we consider how Algorithm 5.3 can be modified to show polynomial time learnability of CQ^{csf} queries of *unbounded arity*. Our focus is the $\text{extract}_{\mathcal{Q}}$ subroutine that may pose a number of membership queries to the teacher that is exponential in the arity of the target query.

Recall that the purpose of $\text{extract}_{\mathcal{Q}}$ is to take as input a CQ q such that $q \subseteq_{\mathcal{O}} q_T$ and then use membership queries to produce a query q' from the class $\mathcal{Q}$ such that $q \subseteq_{\mathcal{O}} q' \subseteq_{\mathcal{O}} q_T$ (see Lemma 5.38). The subroutine is a necessary part of Algorithm 5.3, as hypotheses used in equivalence queries must be from the query class $\mathcal{Q}$. The following example shows that some difficulties are unavoidable when attempting to extract chordal CQs of unbounded arity.

Example 5.40. For the sake of simplicity, suppose we are learning acyclic CQs of arity k and have arrived at the cyclic CQ

$$q(x, \dots, x) \leftarrow r(x, x)$$

such that $q \subseteq_{\varnothing} q_T$ from which we wish to extract an acyclic CQ q'. If we apply an operation like *Expand cycle* to q, then we arrive at the atoms $r(x, x'), r(x', x)$, but it is unclear which occurrences of x in the tuple $(x, \dots, x)$ of answer variables should be replaced with x' to ensure that $q' \subseteq_{\varnothing} q_T$. In the worst case, all 2^k possible answer variable tuples may need to be considered to discover a specific pattern like $q_T(x_1, \dots, x_1, x_2, \dots, x_2) \leftarrow r(x_1, x_2)$.

Therefore, we take a different approach. Instead of requiring that the hypothesis used in equivalence queries must be from $\mathcal{Q}$, we allow equivalence queries with any CQ. Then, we can remove the call to $\text{extract}_{\mathcal{Q}}$ from Algorithm 5.3 and directly use $\mathcal{C}^3_{q_H,\mathcal{O}} \times \mathcal{C}^3_{\mathcal{A},\mathcal{O}}$ as a new hypothesis. This, however, makes it difficult to show that the new hypothesis is closer to q_T. Since q_H is then no longer from the class $\mathcal{Q}$ we cannot apply Lemma 5.28 to show that the sequence of assignments to q_H forms a generalization sequence towards q_T, as in general $q_H \not\subseteq_{\mathcal{O}} \mathcal{C}^3_{q_H,\mathcal{O}}$. This issue can be avoided when learning ELQs, as there is an $\mathcal{EL}$ simulation from $\mathcal{C}^3_{q_H,\mathcal{O}}$ to $\mathcal{U}_{q_H,\mathcal{O}}$ which allows us to apply Lemma 5.26 to show that the new hypothesis is closer to q_T. Unfortunately, while Jung, Lutz, and Wolter characterize ELIQ^{sf} queries in terms of certain simulations [JLW20], no notion of simulation relation for which an equivalent version of Lemma 5.26 holds is known for CQ^{csf} queries.

Hence, in order to guarantee that the new learning algorithm terminates after a polynomial number of steps, we introduce the new subroutine refine, which replaces

the construction of the direct product. Recall that in the loop of Algorithm 5.3, the subroutine $\mathsf{extract}_Q$ takes as input the direct product $q'_H = \mathcal{C}^3_{q_H,\mathcal{O}} \times \mathcal{C}^3_{\mathcal{A},\mathcal{O}}$, and then expands cycles in q'_H, not distinguishing the ABox part and the existentially generated part of the 3-compact models involved. The subroutine refine instead carefully *unravels* the existentially generated part of the two 3-compact models by introducing copies of elements and attaching them in a tree-like manner. A full such unraveling would eventually result in the infinite direct product $\mathcal{U}_{q_H,\mathcal{O}} \times \mathcal{U}_{\mathcal{A},\mathcal{O}}$, but refine interleaves unraveling with calls to the subroutine $\mathsf{minimize}_\mathcal{O}$ and thus obtains a finite initial piece of $\mathcal{U}_{q_H,\mathcal{O}} \times \mathcal{U}_{\mathcal{A},\mathcal{O}}$. Unlike $\mathsf{extract}_Q$, refine does not need to determine new answer variables, as it only creates copies of existential variables. Still, the new hypothesis may have different answer variables to the old one due to the direct product construction.

This description of refine suffices for target queries from $\mathrm{CQ}^{\mathrm{csf}}$ that are rooted. In the general case, disconnected Boolean components might be present or emerge during minimization that are never unraveled by this procedure. To address this, refine subsequently applies $\mathsf{extract}_Q$ to such components, leaving the already unraveled parts untouched. Note that when applied this way, $\mathsf{extract}_Q$ runs in polynomial time, as it is only applied to Boolean subqueries. However, the output of refine is not guaranteed to be a $\mathrm{CQ}^{\mathrm{csf}}$ query, as the initial pieces of $\mathcal{U}_{q_H,\mathcal{O}} \times \mathcal{U}_{\mathcal{A},\mathcal{O}}$ need not be chordal.

Now, we describe the refine subroutine in full detail. It takes as input the ontology $\mathcal{O}$, the old hypothesis $q_H(\overline{x})$ as well as a counterexample $(\mathcal{A}, \overline{a})$ and produces a CQ $q(\overline{x} \otimes \overline{a})$ that is $(q_T, \mathcal{O})$-minimal with $q_H \subseteq_\mathcal{O} q \subseteq_\mathcal{O} q_T$ and $\mathcal{A}, \mathcal{O} \models q(\overline{a})$. For notational convenience, we will view the inputs to refine as two CQs $q_1(\overline{x}_1)$ and $q_2(\overline{x}_2)$, of which we know that $q_i \subseteq_\mathcal{O} q_T$ for $i \in \{1, 2\}$.

First, refine constructs the direct product $\mathcal{C}^3_{q_1,\mathcal{O}} \times \mathcal{C}^3_{q_2,\mathcal{O}}$, views this as a query p with answer variables $\overline{x}_1 \otimes \overline{x}_2$, and then applies $\mathsf{minimize}_\mathcal{O}$ to p. It then incrementally unravels p. All variables of p are pairs (c_1, c_2). Informally, unraveling replaces step-by-step components c_i that are elements of $\Delta^{\mathcal{C}^3_{q_i,\mathcal{O}}} \setminus \mathsf{var}(q_i)$ with corresponding elements of $\Delta^{\mathcal{U}_{q_i,\mathcal{O}}} \setminus \mathsf{var}(q_i)$. In order to make the formal definition and proofs easier to digest, we slightly modify our definition of $\mathcal{U}_{\mathcal{A},\mathcal{O}}$. We now say that $a \rightsquigarrow^r_{\mathcal{A},\mathcal{O}} M$, if $\mathcal{A}, \mathcal{O} \models \exists r. \bigsqcap M(a)$ and ignore the condition that there should be no $b \in \mathsf{ind}(\mathcal{A})$ with $r(a, b) \in \mathcal{A}$ and $\mathcal{A}, \mathcal{O} \models \bigsqcap M(b)$. This means that $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ might contain more traces under this new definition than under the one we used before. This difference in definitions is inessential in the absence of functionality assertions, but makes the following proofs easier.

To now make refine formal, we call $(c_1, c_2) \in \mathsf{var}(p)$ *unraveled* if $c_i \in \Delta^{\mathcal{U}_{q_i,\mathcal{O}}}$ for each $i \in \{1, 2\}$. Note that $(c_1, c_2) \in \mathsf{var}(p)$ and $c_i \notin \Delta^{\mathcal{U}_{q_i,\mathcal{O}}}$ imply that c_i is of the form $c_{a,k,s,A}$. The subroutine refine exhaustively applies the following operation to p:

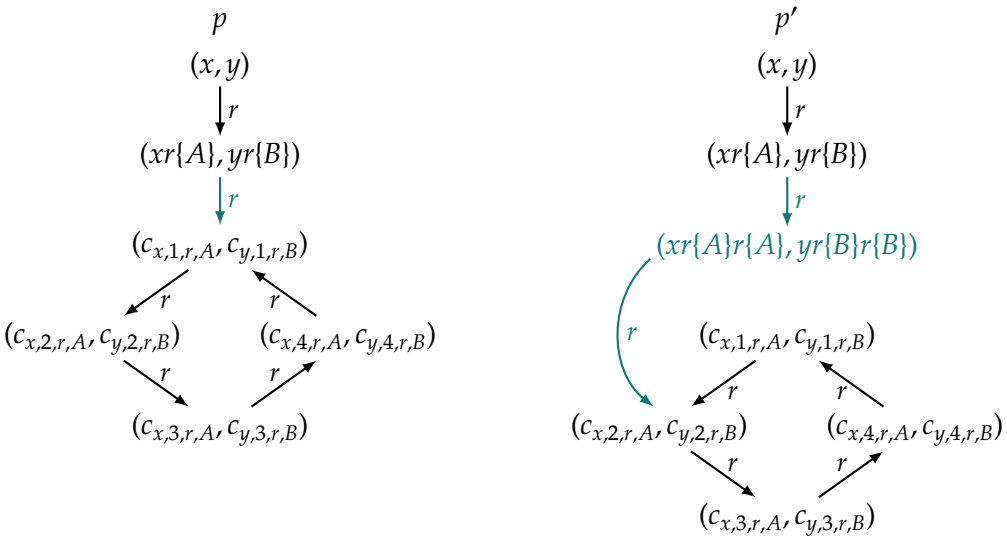

Figure 5.9: An application of *Unravel*.

Unravel. For every atom $r((c_1, c_2), (d_1, d_2)) \in p$ with (c_1, c_2) unraveled and (d_1, d_2) not unraveled, do the following. Remove the atom $r((c_1, c_2), (d_1, d_2))$. For $j \in \{1, 2\}$, set

$$d'_j = \begin{cases} d_j & \text{if } d_j \in \Delta^{\mathcal{U}_{q_j,\mathcal{O}}}, \\ c_j r M & \text{if } d_j = c_{a,k,r,A} \text{ for a set } M \text{ such that } A \in M \text{ and } c_j r M \in \Delta^{\mathcal{U}_{q_j,\mathcal{O}}}. \end{cases}$$

Compensate the removal by adding the following atoms to p:

- $r((c_1, c_2), (d'_1, d'_2))$;
- $A(d'_1, d'_2)$ for all $A(d_1, d_2) \in p$;
- $r((d'_1, d'_2), (e_1, e_2))$ for all $r((d_1, d_2), (e_1, e_2)) \in p$.

Apply $\mathsf{minimize}_{\mathcal{O}}$ to the result.

Note that there may be multiple choices for the set M in the definition of d'_j. If this is the case, *Unravel* makes an arbitrary choice.

We call (d'_1, d'_2) a *copy* of (d_1, d_2). Note that unraveling might introduce several copies of the same original variable (d_1, d_2) and that (d_1, d_2) might or might not be present after unraveling, the latter being the case when $r((c_1, c_2), (d_1, d_2))$ is the only atom that mentions (d_1, d_2).

Example 5.41. Consider the query p displayed in Figure 5.9 that uses as variables elements of $C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}}$ and $\mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}$ for $q_1(x) \leftarrow A(x)$, $q_2(y) \leftarrow B(y)$, and

$\mathcal{O} = \{A \sqsubseteq \exists r.A,\ B \sqsubseteq \exists r.B\}$. It contains the unraveled variable $(xr\{A\}, yr\{B\}) = (c_1, c_2)$ and the not unraveled variable $(c_{x,1,r,A}, c_{y,1,r,B}) = (d_1, d_2)$, as well as the atom $r((c_1, c_2), (d_1, d_2))$. Then, *Unravel* removes $r((c_1, c_2), (d_1, d_2))$ and introduces the new variable $(d'_1, d'_2) = (xr\{A\}r\{A\}, yr\{B\}r\{B\})$, resulting in the query p'.

Due to minimization, at some point all variables in p that can be reached from some element of $\overline{x}_1 \otimes \overline{x}_2$ will be unraveled. However, p might still contain non-unraveled variables, ones that cannot be reached from $\overline{x}_1 \otimes \overline{x}_2$.

Example 5.42. Consider again the ontology $\mathcal{O}$ and query p' from Example 5.41 displayed in Figure 5.9, together with the Boolean target query

$$q_T() \leftarrow r(x_1, x_2) \wedge r(x_2, x_3) \wedge r(x_3, x_4).$$

An application of the subroutine $\mathsf{minimize}_{\mathcal{O}}$ to p' might then remove the variables (x, y), $(xr\{A\}, yr\{B\})$, and $(xr\{A\}ar\{A\}, yr\{B\}r\{B\})$, leaving the cycle consisting of variables $(c_{x,i,r,A}, c_{y,i,r,B})$ for $i \in \{1, \dots, 4\}$ disconnected. This is because the cycle alone suffices in p' to maintain $p' \subseteq_{\mathcal{O}} q_T$. Since only not unraveled variables remain then, *Unravel* will no longer be applied.

To deal with this issue, refine applies $\mathsf{extract}_{\mathcal{Q}}$ to the result of exhaustively applying the *Unravel* operation, with the following modifications:

1. no cycle that involves a variable that is reachable from $\overline{x}_1 \otimes \overline{x}_2$ is considered in the *Expand cycle* operation. As a consequence, the *Expand cycle* operation cannot involve variables in $\overline{x}_1 \otimes \overline{x}_2$, and thus the exponential number of membership queries is avoided;

2. no symmetry that involves a variable that is reachable from $\overline{x}_1 \otimes \overline{x}_2$ is considered in the *Split symmetry* operation.

We now analyze the refine subroutine to verify that it arrives at an initial segment of $\mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}$ after a polynomial number of steps. Let $p_1, p_2, \dots$ be the sequence of queries produced by applying *Unravel* with $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$.

Lemma 5.43. *Let $i \geq 1$. Every cycle in p_i of length at most three consists only of variables from $\mathsf{var}(q_1) \times \mathsf{var}(q_2)$.*

Proof. We prove the lemma by induction on i. In the induction start, recall that $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$. If p_1 contains a cycle $R((x_1, x_2), (x_1, x_2))$ of length 1 with $x_i \notin \mathsf{var}(q_i)$ for some $i \in \{1, 2\}$, then $R(x_i, x_i)$ must be a cycle of length 1 in $C^3_{q_1,\mathcal{O}}$ with $x_i \notin \mathsf{var}(q_i)$ which cannot exist by Lemma 5.24.

Next, if p_1 contains a cycle $R_1((x_1, x_2), (y_1, y_2))$, $R_2((y_1, y_2), (x_1, x_2))$ of length 2, then assume without loss of generality that $x_1 \notin \mathsf{var}(q_1)$. We distinguish cases. If $x_1 = y_1$,

then $R_1(x_1, x_1)$ is a cycle of length 1 in $C^3_{q_1,\mathcal{O}}$, but this is not the case by Lemma 5.24. If $x_1 \neq y_1$, then $R_1(x_1, y_1), R_2(y_1, x_1)$ is a cycle of length 2 in $C^3_{q_1,\mathcal{O}}$ which again contradicts Lemma 5.24.

If p_1 contains a cycle $R_1((x_1, x_2), (y_1, y_2)), R_2((y_1, y_2), (z_1, z_2)), R_3((z_1, z_2), (x_1, x_2))$ of length 3, we can argue similarly that $C^3_{q_1,\mathcal{O}}$ contains a cycle of length 1 or 3 that involves a variable not in $\mathsf{var}(p_i)$, again obtaining a contradiction.

For the induction step, we show that the *Unravel* operation does not create cycles of length 1, 2 or 3. Assume that every cycle in p_i of length at most three consists of only individuals from $\mathsf{var}(q_1) \times \mathsf{var}(q_2)$, and that there is a cycle $R_1((x_1, x_2), (y_1, y_2))$, $R_2((y_1, y_2), (z_1, z_2))$, $R_3((z_1, z_2), (x_1, x_2))$ of length 3 in p_{i+1}. Since $\mathsf{minimize}_{\mathcal{O}}$ only removes variables, one of (x_1, x_2), (y_1, y_2), (z_1, z_2) must be a variable (d'_1, d'_2) introduced in the *Unravel* operation. Replacing (d'_1, d'_2) with its original (d_1, d_2) in the cycle must result in a cycle in p_i which contradicts the induction hypothesis. The same argument can be applied to cycles of length 1 and 2. □

Similar to the *Expand cycle* operation (Lemma 5.35), the *Unravel* operation preserves $\mathcal{EL}$ simulations.

Lemma 5.44. *For all $i \geq 0$, let p'_i be the result of applying* Unravel *to p_i, but before minimization. Then,*

1. *$p'_i, x \preceq_{\mathcal{EL}} p_i, x$ and $p_i, x \preceq_{\mathcal{EL}} p'_i, x$ for all $x \in \mathsf{var}(p_i) \cap \mathsf{var}(p'_i)$*
2. *$p'_i, x' \preceq_{\mathcal{EL}} p_i, x$ and $p_i, x \preceq_{\mathcal{EL}} p'_i, x'$ for all copies $x' \in \mathsf{var}(p'_i) \setminus \mathsf{var}(p_i)$ of some $x \in \mathsf{var}(p_i)$.*

Proof. Define a relation $S \subseteq \mathsf{var}(p_i) \times \mathsf{var}(p'_i)$ by taking:

- $(x, x) \in S$ for all $x \in \mathsf{var}(p_i) \cap \mathsf{var}(p'_i)$, and
- $(x, x') \in S$ for all copies $x' \in \mathsf{var}(p'_i)$ of some element $x \in \mathsf{var}(p_i)$.

S is a simulation from p_i to p'_i, and its inverse S^- is a simulation from p'_i to p_i. □

The next lemma is the most intricate to prove in the analysis of *Unravel*.

Lemma 5.45. *For all $i \geq 1$, $p_i \subseteq_{\mathcal{O}} q_T$.*

Proof. We prove the lemma by induction on i. The induction start is immediate since $q_T(\overline{x}_0) \to \mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}, \overline{x}_1 \otimes \overline{x}_2$, and there is a homomorphism from $\mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}$ to $C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}}$ that is the identity on $\overline{x}_1 \otimes \overline{x}_2$. Thus, for $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$, $p_1 \subseteq_{\mathcal{O}} q_T$ by Lemma 4.20.

For the induction step, let p be the result of applying *Unravel* to p_i but before minimization and assume that there is a homomorphism h from q_T to $\mathcal{U}_{p_i,\mathcal{O}}$ with

$h(\overline{x}_0) = \overline{x}_1 \otimes \overline{x}_2$. Let U be the set of all variables $(d_1, d_2) \in \mathsf{var}(p_i)$ such that some atom $r((c_1, c_2), (d_1, d_2))$ was removed by *Unravel*. Note that if $(d_1, d_2) \in U$, then $(d_1, d_2) \notin \mathsf{var}(q_1) \times \mathsf{var}(q_2)$. In what follows, we construct a homomorphism g from q_T to $\mathcal{U}_{p,\mathcal{O}}$ with $g(\overline{x}_0) = \overline{x}_1 \otimes \overline{x}_2$. Thus, $p \subseteq_{\mathcal{O}} q_T$, which is preserved by $\mathsf{minimize}_{\mathcal{O}}$, meaning that $p_{i+1} \subseteq_{\mathcal{O}} q_T$, as desired.

For this, first observe that if $R((c_1, c_2), (d_1, d_2))$ is an atom in p_i with (c_1, c_2) unraveled and (d_1, d_2) not unraveled, then R is a role name, but not an inverse role. For a variable x in q_T, let us denote with V_x the set of all atoms $R(x, y) \in q_T$ such that $h(y) \in \mathsf{var}(p_i)$ and $h(y)$ is unraveled. We observe the following about V_x.

Claim 1. Let $x \in \mathsf{var}(q_T)$ such that $h(x) = (d_1, d_2) \in U$. Then, there is a role name r such that all atoms in V_x are of shape $r(y, x)$ and one of the following is the case:

i. V_x is a singleton;

ii. d_1 has the form $c_{z,0,r,A}$ and for every $r(y, x) \in V_x$, q_2 contains an atom $r(z', d_2)$ with $h(y) = (z, z')$;

iii. d_2 has the form $c_{z,0,r,A}$ and for every $r(y, x) \in V_x$, q_1 contains an atom $r(z', d_1)$ with $h(y) = (z', z)$;

iv. d_1 has the form $c_{z_1,0,r,A_1}$, d_2 has the form $c_{z_2,0,r,A_2}$, and $h(y) = (z_1, z_2)$ for every $r(y, x) \in V_x$.

Proof of Claim 1. To show the first part, let $R(y_1, x), S(y_2, x) \in V_x$. Since $h(x) = (d_1, d_2)$ is not unraveled, but $h(y_1)$ and $h(y_2)$ are unraveled, R and S are role names. Moreover, (d_1, d_2) not being unraveled means that at least one of the d_j takes the shape $c_{z,k,r,A}$ for some role name r. By definition of $\mathcal{C}^3_{q_j,\mathcal{O}}$, for every $s(d, c_{a,k,r,A}) \in \mathcal{C}^3_{q_j,\mathcal{O}}$, we have $s = r$. Hence, for every $s((c_1, c_2), (d_1, d_2)) \in p_i$ we have $s = r$ as well. Thus, $R = S = r$ and all atoms in V_x are based on the same role name r.

Now for the second part. Assume that Case i does not apply. Then we find $r(y_1, x), r(y_2, x) \in V_x$ with $y_1 \neq y_2$. Since q_T is symmetry-free, $r(y_1, x), r(y_2, x)$ must be a safe symmetry. Since $h(x) \in U$, x is not an answer variable. The fact that $h(y_1)$ is unraveled, but $h(x)$ is not, implies that $\mathsf{dist}(z, x) + 1 \neq \mathsf{dist}(z, y_1)$ for any answer variable z. Additionally, q_T may contain no atom of the form $s(x, x)$, since no atom $s(h(x), h(x))$ may exist in p by Lemma 5.43 and $h(x)$ not being unraveled. The remaining possibilities for $r(y_1, x), r(y_2, x)$ to be a safe symmetry are: for $j \in \{1, 2\}$, the atom $r(y_j, x)$ occurs on a cycle, or there is an atom $s(y_j, y_j) \in q_T$. Without loss of generality, assume $j = 1$. If $s(y_1, y_1) \in q_T$, then it follows that $h(y_1) = h(y_2) = (z_1, z_2) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$. If $r(y_1, x)$ occurs on a cycle C in q_T, then by chordality of q_T, it follows that C has at most length three. Since h is a homomorphism from q_T to $\mathcal{U}_{p_i,\mathcal{O}}$, the h-image of C in p_i must contain a cycle of length at most three. By Lemma 5.43, this

implies that $h(x)$ is not part of a cycle in the h-image of C. Consequently, the cycle C has to be of the shape

$$r(y_1, x),\ r^-(x, z),\ s(y_1, z)$$

with $h(y_1) = h(z) = (z_1, z_2) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$.

It follows that $i = 1$, since if $i > 1$ all successors of elements of $\mathsf{var}(q_1) \times \mathsf{var}(q_2)$ are unraveled. We distinguish the following cases:

- $d_1 \in \mathsf{var}(q_1)$ and $d_2 \in \mathsf{var}(q_2)$.

  This is impossible because (d_1, d_2) is not unraveled.

- d_1 has shape $c_{z_1,0,r,A}$ and $d_2 \in \mathsf{var}(q_2)$.

  Then z_1 is the unique r-predecessor of d_1 in $C^3_{q_1,\mathcal{O}}$ that can appear in the first component of an unraveled element. Let $r(y, x) \in V_x$. Then $h(y) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$ because $h(y)$ is unraveled and $i = 1$. Since $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$, $r(y, x) \in V_x$ thus implies that there is an atom $r(z', d_2) \in q_2$ such that $h(y) = (z_1, z')$. Thus, we are in Case ii.

- d_2 has shape $c_{z_2,0,r,A}$ and $d_1 \in \mathsf{var}(q_1)$.

  We argue as in the previous case, but end up in Case iii.

- d_1 has shape $c_{z_1,0,r,A_1}$ and d_2 has shape $c_{z_2,0,r,A_2}$. By definition of the models $C^3_{q_i,\mathcal{O}}$ and since in $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$, (z_1, z_2) is the unique unraveled r-predecessor of (d_1, d_2) in $p_i = p_1$. Thus, we are in Case iv.

This completes the proof of Claim 1.

Next, with every variable $x \in \mathsf{var}(q_T)$ such that $h(x) \in \mathsf{var}(p_i)$, we associate a set Z_x that consists of all variables $y \in \mathsf{var}(q_T)$ such that q_T contains a simple path $R_1(z_1, z_2), \dots, R_{m-1}(z_{m-1}, z_m)$ from x to y where $h(z_2), \dots, h(z_m)$ are all proper traces in $\mathcal{U}_{p_i,\mathcal{O}}$ that start with $h(x)$. We observe the following about the sets Z_x.

Claim 2. For all $y \in \mathsf{var}(q_T)$ with $h(y) \notin \mathsf{var}(p_i)$, there is at most one $x \in \mathsf{var}(q_T)$ with $y \in Z_x$ and $h(x) \in U$.

Proof of Claim 2. Suppose that $y \in \mathsf{var}(q_T)$ with $h(y) \notin \mathsf{var}(p_i)$ and that there are distinct variables $x_1, x_2 \in \mathsf{var}(q_T)$ with $y \in Z_{x_j}$ and $h(x_j) \in U$ for $j \in \{1, 2\}$. Let

$$\pi_1 = R_1(z_1, z_2), \dots, R_n(z_{n-1}, z_n) \quad \text{and}$$
$$\pi_2 = S_1(z'_1, z'_2), \dots, S_m(z'_{m-1}, z'_m)$$

be paths in q_T from x_1 to y and from x_2 to y, respectively, such that $h(z_j) \neq h(x_1)$ for all $j \in \{1, \dots, n\}$ and $h(z'_j) \neq h(x_2)$ for all $j \in \{1, \dots, m\}$. Note that h is a homomorphism

from π_j to the trace subtree of $\mathcal{U}_{p_i,\mathcal{O}}$ rooted at $h(x_j)$, for $j \in \{1,2\}$. Since $h(y)$ is both in the subtree below $h(x_1)$ and below $h(x_2)$, it follows that $h(x_1) = h(x_2)$.

We analyze the structure of the paths π_1 and π_2. Let us first verify that all R_j and all S_j are role names. We do this explicitly only for the R_j. Let $\mathcal{I}$ denote the subtree of $\mathcal{U}_{p_i,\mathcal{O}}$ rooted at $h(x_1)$, that is, the restriction of $\mathcal{U}_{p_i,\mathcal{O}}$ to all traces that start with $h(x_1)$, including $h(x_1)$ itself. By construction of $\mathcal{U}_{p_i,\mathcal{O}}$, $\mathcal{I}$ is a directed tree. Then R_1 must be a role name since $(h(x_1), h(z_1)) \in R_1^{\mathcal{I}}$, $h(x_1)$ is the root of $\mathcal{I}$, and $h(z_1)$ in $\Delta^{\mathcal{I}}$. Now, let ℓ be minimal such that R_ℓ is an inverse role r^- and consider the atoms $R_{\ell-1}(z_{\ell-1}, z_\ell), r^-(z_\ell, z_{\ell+1})$ in q_T. Since h is a homomorphism and $\mathcal{I}$ is a directed tree, we know that $R_{\ell-1} = r$, and thus there are atoms $r(z_{\ell-1}, z_\ell), r(z_{\ell+1}, z_\ell)$ in q_T.

If $z_{\ell-1} \neq z_{\ell+1}$, then q_T contains a symmetry, which must be safe since q_T is symmetry-free. If there is a single answer variable $\hat{z}$ of q_T and $\mathsf{dist}(\hat{z}, z_\ell) + 1 = \mathsf{dist}(\hat{z}, z_{\ell-1})$, then, since $h(z_\ell)$ is a trace, there must be an atom $r(z_{\ell'}, z_\ell)$ as the last element of the path from $\hat{z}$ to z_ℓ with $z_{\ell'} \neq z_{\ell-1}$. Then either $h(z_{\ell'}) \in U$ or $h(z_{\ell'})$ is a trace, and we continue this argument with the symmetry $r(z_{\ell'}, z_\ell), r(z_{\ell-1}, z_\ell)$. Since $h(z_\ell)$ is a trace, z_ℓ is not an answer variable and there is no atom $s(z_\ell, z_\ell)$. Furthermore, since $h(z_{\ell-1})$ and $h(z_{\ell+1})$ are either traces or elements of U (if $z_{\ell-1} = x_1$), there is no atom $s(z_{\ell-1}, z_{\ell+1})$ or $s(z_{\ell-1}, z_{\ell+1})$. Hence, one of the two atoms $r(z_{\ell-1}, z_\ell), r(z_{\ell+1}, z_\ell)$ occurs on a cycle C in q_T.

Let us assume that this is atom $r(z_{\ell-1}, z_\ell)$, the case of atom $r(z_{\ell+1}, z_\ell)$ is analogous. Since q_T is chordal, we can assume that C has length at most three. Since h is a homomorphism from q_T to $\mathcal{U}_{p_i,\mathcal{O}}$, the image of C contains a cycle C' of length at most three in $\mathcal{U}_{p_i,\mathcal{O}}$. Even if h is not injective, the cycle C' must contain $h(z_\ell)$ or $h(z_{\ell-1})$. However, both possibilities lead to a contradiction. If C' contains $h(z_\ell)$, then $h(z_\ell) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$ by Lemma 5.43, but this is not the case since $h(z_\ell)$ is in $\mathcal{I}$ and different from $h(x_1)$. If C' contains $h(z_{\ell-1})$, then $h(z_{\ell-1})$ must be $h(x_1)$, and C' witnesses that $h(x_1) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$, in contradiction to $h(x_1) \in U$.

Therefore, $z_{\ell-1} = z_{\ell+1}$ and we can drop the two atoms from the path. At this point, we have established that all R_j and S_j are role names r_j, s_j. Since $\mathcal{I}$ is a directed tree, it follows that $m = n$ and $r_j = s_j$ for all j. Since $z_1 \neq z_1'$ and $z_n = z_m'$, there is some $\ell > 0$ such that $z_\ell = z_\ell'$, $z_{\ell-1} \neq z_{\ell-1}'$. But then q_T contains a symmetry $r_\ell(z_{\ell-1}, z_\ell), r_\ell(z_{\ell-1}', z_\ell)$. This leads to a contradiction using the same argument as above. This completes the proof of Claim 2.

Using the sets V_x and Z_x, we now define the desired homomorphism g in four stages.

1. Define $g(x) = h(x)$ for all $x \in \mathsf{var}(q_T)$ such that $h(x) \in \mathsf{var}(p_i) \setminus U$ or $h(x)$ is a trace starting with some variable not in U.

2. For every $x \in \mathsf{var}(q_T)$ with $h(x) = (d_1, d_2) \in U$, we distinguish cases according to Claim 1:

   a) If $V_x = \emptyset$, then define $g(x) = h(x)$. We argue that this is well-defined, that is, $h(x) \in \mathsf{var}(p)$. Suppose to the contrary that $h(x) \notin \mathsf{var}(p)$. By definition of *Unravel*, this can only be the case if p_i contains only a single assertion that mentions $h(x)$ and this assertion is of shape $r((c_1, c_2), h(x))$ with (c_1, c_2) unraveled. Since x has to occur in some atom in q_T and h is a homomorphism, x occurs in an atom $r(z, x) \in q_T$ such that $h(z) = (c_1, c_2)$. Hence, $r(z, x) \in V_x \neq \emptyset$, a contradiction.

   b) If Case i applies and $V_x = \{r(y, x)\}$, define $g(x)$ to be the copy (d_1', d_2') of (d_1, d_2) introduced by the *Unravel* operation for $r(h(y), h(x)) \in p_i$.

   c) If $V_x \neq \emptyset$ and Case ii applies (but Case i does not), then define $g(x)$ to be the copy (zrM, d_2) of (d_1, d_2) for a set M with $A \in M$ such that zrM is a trace, where z, A are as in Case ii of Claim 1.

   d) If $V_x \neq \emptyset$ and Case iii applies (but Case i does not), analogously define $g(x)$ to be the copy (d_1, zrM).

   e) If $V_x \neq \emptyset$ and Case iv applies (but Case i does not), define $g(x)$ to be the copy $(z_1 r M_1, z_2 r M_2)$ where M_i are sets with $A_i \in M_i$ such that $z_1 r M_1$ and $z_2 r M_2$ are traces with z_1, z_2, A_1, A_2 as in Case iv.

3. For every x with $h(x) \in U$ and every $y \in Z_x$, $h(y)$ is a trace that starts with $h(x)$. Define $g(y)$ to be the same trace, but with the first element $h(x)$ replaced by $g(x)$. Using Lemma 5.44, it can be verified that $g(y)$ is indeed an element in $\mathcal{U}_{p,\mathcal{O}}$ using the fact that the trace subtrees below $g(x)$ and $h(x)$ in $\mathcal{U}_{p,\mathcal{O}}$ and $\mathcal{U}_{p_i,\mathcal{O}}$, respectively, are identical.

4. For every $(d_1, d_2) \in U$ and $y \in \mathsf{var}(q_T)$ such that $h(y)$ is a trace that starts with (d_1, d_2) and $y \notin Z_x$ for all x with $h(x) \in U$, choose some copy (d_1', d_2') of (d_1, d_2) and define $g(y)$ to be the trace $h(y)$ with the first element (d_1, d_2) replaced by (d_1', d_2').

The four stages above define $g(x)$ for all $x \in \mathsf{var}(q_T)$. It remains to verify that g is a homomorphism from q_T to $\mathcal{U}_{p,\mathcal{O}}$ with $g(\overline{x}_0) = \overline{x}_1 \otimes \overline{x}_2$. Observe that $h(x) \in \mathsf{var}(q_1) \times \mathsf{var}(q_2)$ for every $x \in \overline{x}_0$ and that $U \cap (\mathsf{var}(q_1) \times \mathsf{var}(q_2)) = \emptyset$. Thus, Stage 1 of the definition of g implies $g(\overline{x}_0) = h(\overline{x}_0)$.

Now, let $A(x) \in q_T$ and thus $h(x) \in A^{\mathcal{U}_{p_i,\mathcal{O}}}$. We distinguish the following cases:

- If $g(x)$ was defined in Stage 1, then $g(x) = h(x)$. First, assume that $h(x) \in \mathsf{var}(p_i)$. Then $p_i, h(x) \preceq_{\mathcal{EL}} p, g(x)$ by Lemma 5.44 and thus $\mathcal{U}_{p_i,\mathcal{O}}, h(x) \preceq_{\mathcal{EL}} \mathcal{U}_{p,\mathcal{O}}, g(x)$.

Hence, $g(x) \in A^{\mathcal{U}_{p,\mathcal{O}}}$ by Lemma 5.26. Now assume that $h(x) \notin \mathsf{var}(p_i)$. Then $h(x) = g(x)$ is a trace, and traces in $\mathcal{U}_{p_i,\mathcal{O}}$ and $\mathcal{U}_{p,\mathcal{O}}$ that end with the same set of concepts M satisfy the same concept names.

- If $g(x)$ was defined in Stage 2, then $g(x) = (d_1', d_2')$ is a copy of $h(x) = (d_1, d_2)$ or $g(x) = h(x)$. By Lemma 5.44, we have $p_i, h(x) \preceq_{\mathcal{EL}} p, g(x)$, and thus $g(x) \in A^{\mathcal{U}_{p,\mathcal{O}}}$ by Lemma 5.26.
- If $g(x)$ was defined in Stage 3 or 4, then $h(x)$ and $g(x)$ are both traces that end with the same set of concepts M and, by construction of universal models, thus satisfy the same concept names. Consequently, $h(x) \in A^{\mathcal{U}_{p_i,\mathcal{O}}}$ implies $g(x) \in A^{\mathcal{U}_{p,\mathcal{O}}}$.

Finally, let $r(x, y) \in q_T$ and thus $(h(x), h(y)) \in r^{\mathcal{U}_{p_i,\mathcal{O}}}$. We distinguish the following cases:

- It cannot be that both $h(x)$ and $h(y)$ are elements of U, by definition of the *Unravel* operation.
- If both $h(x)$ and $h(y)$ are not elements of U, then both $g(x)$ and $g(y)$ were defined in the same stage, one of Stage 1, 3, and 4. We can then argue analogously to the case of concept atoms that $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$.
- If $h(x) = (d_1, d_2) \in U$ and $h(y) \notin U$, then we distinguish cases:
  - If $h(y) \notin \mathsf{var}(p_i)$, then $h(y)$ is a trace of the form $(d_1, d_2)rM$ in $\mathcal{U}_{p_i,\mathcal{O}}$. Thus, $g(y)$ was defined in Stage 3 as a trace $(d_1', d_2')rM$ for $g(x) = (d_1', d_2')$. It follows that $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$.
  - If $h(y) \in \mathsf{var}(p_i)$ is not unraveled, then by definition of *Unravel*, it is the case that $r((d_1', d_2'), h(y)) \in p$ for all copies (d_1', d_2') of (d_1, d_2), and $r((d_1, d_2), h(y)) \in p$. We know that $g(x)$ was defined in Stage 2 and is either $h(x)$ or some copy thereof, and $h(y)$ was defined in Stage 1, thus $g(y) = h(y)$. Consequently, $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$.
  - It cannot be the case that $h(y) \in \mathsf{var}(p_i)$ is unraveled. By Claim 1, S is a role name for every atom $S(z, x) \in q_T$ such that $h(z)$ is unraveled. However, this is not the case for the atom $r^-(y, x) \in q_T$ we started with.
- If $h(x) \notin U$ and $h(y) = (d_1, d_2) \in U$, then $h(x) \in \mathsf{var}(p_i)$ since $(h(x), h(y)) \in r^{\mathcal{U}_{p_i,\mathcal{O}}}$ and by definition of universal models. We distinguish cases according to Claim 1:
  - If $V_y = \varnothing$, then $g(y) = h(y)$, by Stage 2a. Moreover, as $h(x) \in \mathsf{var}(p_i) \setminus U$, we have $g(x) = h(x)$, by Stage 1. Hence, $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$.

- If Case i applies and $V_y = \{r(x, y)\}$ with $h(x)$ unraveled, then $g(y)$ was defined in Stage 2b and $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$ by definition of the *Unravel* operation.
- If Case ii applies to V_y, then d_1 has the form $c_{z,0,r,A}$ and for every $r(y', y) \in V_y$, q_2 contains an atom $r(z', d_2)$ with $h(y') = (z, z')$. Moreover, $g(y)$ was defined in Stage 2c and $g(y) = (zrM, d_2)$ for a set M with $A \in M$.
  - If $h(x)$ is unraveled, then $h(x) = g(x) = (z, z')$. By definition of *Unravel*, $r((z, z'), (zrM, d_2)) \in p$. Hence, $(g(x), g(y)) \in r^{\mathcal{U}_{p,\mathcal{O}}}$.
  - If $h(x)$ is not unraveled, then it was defined in Stage 1 and $h(x) = g(x)$. By definition of *Unravel*, $r(h(x), (zrM, d_2)) \in p$.
- If Case iii applies to V_y, the argument is symmetric.
- If Case iv applies to V_y, d_1 has the form $c_{z_1,0,r,A_1}$, d_2 has the form $c_{z_2,0,r,A_2}$, $h(x) = (z_1, z_2)$, and $g(y)$ was defined in Stage 2e to be $(z_1 r M_1, z_2 r M_2)$ for sets M_j with $A_j \in M_j$. Since $h(x)$ is unraveled, $g(x)$ was set to $h(x)$ in Stage 1, and the definition of *Unravel* implies $r(g(x), g(y)) \in p$. □

We can now use the bound on generalization sequences to show that refine terminates in polynomial time.

Lemma 5.46. *Let $q_1(\overline{x}_1)$ and $q_2(\overline{x}_2)$ be CQs with $q_1 \subseteq_{\mathcal{O}} q_T$ and $q_2 \subseteq_{\mathcal{O}} q_T$. Then, the subroutine* refine$(\mathcal{O}, q_1, q_2)$ *runs in time polynomial in $\|q_T\| + \|q_1\| + \|q_2\| + \|\mathcal{O}\|$ and returns a $(q_T, \mathcal{O})$-minimal CQ p such that $q_1 \subseteq_{\mathcal{O}} p$, $q_2 \subseteq_{\mathcal{O}} p$, and $p \subseteq_{\mathcal{O}} q_T$.*

Proof. Consider again the sequence $p_1, p_2, \ldots$ produced by applying the *Unravel* operation to $p_1 = \mathsf{minimize}_{\mathcal{O}}(C^3_{q_1,\mathcal{O}} \times C^3_{q_2,\mathcal{O}})$. We argue that the length of this sequence is bounded by $|\mathsf{var}(q_T)| + 1$. Indeed, the following can be shown by induction on i.

Claim. Let $i \geq 1$. Then every variable y in $\mathsf{var}(p_i)$ with $\mathsf{dist}(y, x) \leq i - 1$ for some $x \in \overline{x}_1 \otimes \overline{x}_2$ is unraveled.

Since every p_i is $(q_T, \mathcal{O})$-minimal, and therefore $|\mathsf{var}(p_i)| \leq |\mathsf{var}(q_T)|$, it follows that the *Unravel* operation is no longer applicable to $p_{|\mathsf{var}(q_T)|+2}$.

Next, we observe that the application of $\mathsf{extract}_{\mathcal{Q}}$ to components of p that are not connected to any element of $\overline{x}_1 \otimes \overline{x}_2$ is bounded by the same arguments as in Section 5.3. Due to the properties of *Expand cycle* stated in Lemma 5.36 and properties of *Split symmetry* stated in Lemma 5.37, the sequence of produced queries forms a generalization sequence of $(q_T, \mathcal{O})$-minimal CQs towards q_T. Theorem 5.32 then implies a polynomial bound on the length of the sequence. Therefore, refine terminates after polynomially many steps and membership queries.

Algorithm 5.4: The modified learning algorithm for $\mathrm{CQ}^{\mathrm{csf}}$ under $\mathcal{EL}^r$ ontologies

Input A signature Σ, an $\mathcal{EL}^r$ ontology $\mathcal{O}$ in normal form and an arity k.
Output A k-ary $q_H \in \mathrm{CQ}^{\mathrm{csf}}$ such that $q_H \equiv_{\mathcal{O}} q_T$

$q_H^0 :=$ initial-CQ$(\Sigma, \mathcal{O}, k)$
$q_H := q_H^0$
while the CQ-equivalence query "$q_H \equiv_{\mathcal{O}} q_T$?" returns a counterexample $(\mathcal{A}, \overline{a})$ **do**
    $q_H :=$ refine$(\mathcal{O}, q_H, (\mathcal{A}, \overline{a}))$
end while
return q_H

It remains to show that for $p(\overline{x}) =$ refine$(\mathcal{O}, q_1, q_2)$, $q_1 \subseteq_{\mathcal{O}} p$, $q_2 \subseteq_{\mathcal{O}} p$ and $p \subseteq_{\mathcal{O}} q_T$. The last query implication follows from Lemma 5.45, Lemma 5.36, Lemma 5.37 and the fact that minimize$_{\mathcal{O}}$ preserves query implication of q_T.

To show the first query implication, let p' be the restriction of p to variables that are reachable from an element of $\overline{x}_1 \otimes \overline{x}_2$, and let p'' be the restriction of p to all variables that are not reachable. Hence, $p = p' \cup p''$. Since *Unravel* was applied exhaustively, all variables in p' are unraveled. Therefore, $\mathsf{var}(p') \subseteq \Delta^{\mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}}$. Using the definition of $\mathcal{C}^3$ and *Unravel*, it is easy to see that the identity is a homomorphism h' from p to $\mathcal{U}_{q_1,\mathcal{O}} \times \mathcal{U}_{q_2,\mathcal{O}}$ with $h'(\overline{x}) = \overline{x}_1 \otimes \overline{x}_2$. Projection to the left components yields a homomorphism that witnesses $q_1 \subseteq_{\mathcal{O}} p'$.

For the second part p'', note that no variable in p'' is unraveled. That is, every variable is either an element of $(\Delta^{\mathcal{C}^3_{q_1,\mathcal{O}} \times \mathcal{C}^3_{q_2,\mathcal{O}}}) \setminus (\mathsf{var}(q_1) \times \mathsf{var}(q_2))$, or a copy of such an element introduced by *Expand cycle* or *Split symmetry*. The natural mapping h'' that maps copies to their originals, is then a homomorphism from p'' to $\mathcal{C}^3_{q_1,\mathcal{O}} \times \mathcal{C}^3_{q_2,\mathcal{O}}$. Projection to the left component yields a homomorphism g'' from p'' to $\mathcal{C}^3_{q_1,\mathcal{O}}$. Since *Expand cycle* and *Split symmetry* were applied exhaustively to p'', p'' must be chordal and symmetry free. It then follows from Lemma 5.28 that $q_1 \subseteq_{\mathcal{O}} p''$.

The second implication can be shown in the same manner. □

The modified version of Algorithm 5.3 that uses refine and CQ-equivalence queries is displayed as Algorithm 5.4. It is important to note that although Algorithm 5.4 may during its run produce hypotheses q_H that are not chordal and symmetry free, only $\mathrm{CQ}^{\mathrm{csf}}$ queries will be returned.

Lemma 5.47. *Let $q(\overline{x}_1), p(\overline{x}_2)$ be CQs. If p is chordal and symmetry-free, $q \equiv_{\mathcal{O}} p$ and q is $(p, \mathcal{O})$-minimal, then p is also chordal and symmetry-free.*

Proof sketch. Let h be a homomorphism from p to $\mathcal{U}_{q,\mathcal{O}}$ with $h(\overline{x}_2) = \overline{x}_1$ and g a

homomorphism from q to $\mathcal{U}_{p,\mathcal{O}}$ with $h(\overline{x}_1) = \overline{x}_2$. Note that due to $(p, \mathcal{O})$-minimality of q, g must be injective, as we could otherwise obtain a non-injective homomorphism from q to $\mathcal{U}_{q,\mathcal{O}}$ by composing g with an extension of h. Then, a variant of the proof of Point 2 of Lemma 5.36 shows that the composition of g and h must map every cycle in q injectively onto a cycle in q. Since no chordless cycle of length at least 4 exists in p, it follows that no such cycle can exist in q.

A similar argument applies to non-safe symmetries. The composition of g and h must map symmetries to symmetries. A variant of the proof of Lemma 5.37 shows that a non-safe symmetry can be mapped to a safe symmetry by the composition of g and h. Since no non-safe symmetries exist in p, it follows that only safe symmetries exist in q. □

It remains to show that Algorithm 5.4 terminates after a polynomial number of iterations. Similar to the previous proof for Algorithm 5.3, we argue that the sequence of assignments to q_H forms a generalization sequence of $(q_T, \mathcal{O})$-minimal CQs towards q_T under $\mathcal{O}$. The polynomial bound on the length of such sequences in Theorem 5.32 then yields the desired bound on the number of iterations.

Theorem 5.48. $\mathrm{CQ}^{\mathrm{csf}}$ *queries are polynomial time learnable under* $\mathcal{EL}^r$ *ontologies using membership queries and CQ-equivalence queries.*

Proof. Let $q_1, q_2, \ldots$ be the sequence of assignments to q_H during a run of Algorithm 5.4. Lemma 5.46 implies that the refine subroutine runs in polynomial time in its inputs and of q_T, and that each q_i is $(q_T, \mathcal{O})$-minimal. Therefore, $|\mathsf{var}(q_i)| \leq |\mathsf{var}(q_T)|$ for all $i \geq 0$. Hence, it suffices to show that the sequence $q_1, q_2, \ldots$ forms a generalization sequence towards q_T under $\mathcal{O}$.

First, we show that $q_i \subseteq_{\mathcal{O}} q_T$ for all $i \geq 1$. We show this by induction on i. For $i = 1$ this is immediate, since $q_1 = q_H^0$. Now assume that $q_{i-1} \subseteq_{\mathcal{O}} q_T$. The counterexample $(\mathcal{A}, \overline{a})$ returned by the equivalence query "$q_{i-1} \equiv_{\mathcal{O}} q_T$" must then be such that $\mathcal{A}, \mathcal{O} \models q_T(\overline{a})$ and $\mathcal{A}, \mathcal{O} \not\models q_{i-1}(\overline{a})$. If we view $(\mathcal{A}, \overline{a})$ as a CQ $q_{\mathcal{A}}$, this implies that $q_{\mathcal{A}} \subseteq_{\mathcal{O}} q_T$ and $q_{\mathcal{A}} \not\subseteq_{\mathcal{O}} q_{i-1}$. Hence, $q_i \subseteq_{\mathcal{O}} q_T$ by Lemma 5.46, as required.

Next, we show that $q_{i-1} \subseteq_{\mathcal{O}} q_i$ and $q_i \not\subseteq_{\mathcal{O}} q_{i-1}$ for all $i \geq 2$. The first point follows directly from Lemma 5.46 and the fact that $q_i = \mathsf{refine}(q_{i-1}, (\mathcal{A}, \overline{a}))$. The second point follows from $q_{\mathcal{A}} \not\subseteq_{\mathcal{O}} q_{i-1}$ and $q_{\mathcal{A}} \subseteq_{\mathcal{O}} q_i$.

Therefore, the sequence $q_1, q_2, \ldots$ forms a generalization sequence towards q_T under $\mathcal{O}$. Since all q_i are $(q_T, \mathcal{O})$-minimal, Theorem 5.32 implies a polynomial bound on the length of the sequence. Algorithm 5.4 must terminate after a number of iterations that is polynomial in $|\Sigma|$, $\|\mathcal{O}\|$ and $\|q_T\|$. □

Note that in contrast to Algorithm 5.3, Algorithm 5.4 uses CQ-equivalence queries and thus requires a more capable teacher. This allows us to learn $\mathrm{CQ}^{\mathrm{csf}}$ queries of any arity in polynomial time.

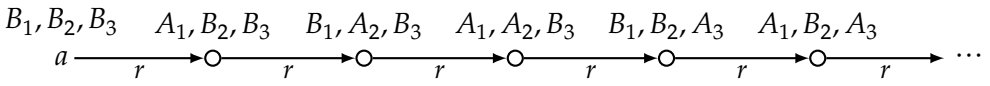

Figure 5.10: The initial segment of the universal model $\mathcal{U}_{\mathcal{A}_3,\mathcal{O}_3}$ of the $\mathcal{ELI}$ ontology $\mathcal{O}_3$ and the ABox $\mathcal{A}_3$, which contains a 3-bit binary counter.

5.5 Learning under $\mathcal{ELI}$ ontologies

In the previous sections, we have looked at polynomial time learnability of queries under $\mathcal{EL}^r$ ontologies. $\mathcal{EL}^r$ allows for polynomial time reasoning, but its expressiveness is limited in several ways. One of these limitations is that inverse roles can only be used as part of *range restrictions*, that is concept inclusions of the form $\exists r^-.\top \sqsubseteq A$, where r is a role name and A is a concept name. In this section, we consider learnability of queries under ontologies written in $\mathcal{ELI}$, that allows the unrestricted use of inverse roles. For example, $\exists r^-.A \sqsubseteq \exists s^-.B$ is an $\mathcal{ELI}$ concept inclusion that cannot be expressed in $\mathcal{EL}^r$.

The polynomial time learnability results in Section 5.3 and Section 5.4 rely on two crucial properties of $\mathcal{EL}^r$: First, that there exist polynomial size CQ^{csf}-universal models of $\mathcal{EL}^r$ ontologies that can be computed in polynomial time, and second, that whether $\mathcal{A}, \mathcal{O} \models A(a)$ can be decided in polynomial time. Both of these properties no longer hold for $\mathcal{ELI}$ ontologies. Indeed, it is known that the standard reasoning problems for $\mathcal{ELI}$ are ExpTime-complete [BBL08] and that there are $\mathcal{ELI}$ ontologies for which no polynomial size ELQ-universal models exist.

Example 5.49. For some $n \geq 1$, let $\mathcal{O}_n$ be the $\mathcal{ELI}$ ontology from Example 4.36 that contains the following concept inclusions for all i with $1 \leq i \leq n$ and for all j with $1 \leq j < i$:

$$B_i \sqsubseteq \exists r.\top$$

$$\exists r^-.(A_1 \sqcap \cdots \sqcap A_{i-1} \sqcap B_i) \sqsubseteq A_i \qquad \exists r^-.(A_1 \sqcap \cdots \sqcap A_{i-1} \sqcap A_1) \sqsubseteq B_i$$

$$\exists r^-.B_i \sqcap B_j \sqsubseteq B_i \qquad \exists r^-.A_i \sqcap B_j \sqsubseteq A_i$$

Together with the ABox $\mathcal{A}_n = \{B_1(a), \dots, B_n(a)\}$, these concept inclusion generate an r-path of length 2^n in $\mathcal{U}_{\mathcal{A}_n,\mathcal{O}_n}$, which is displayed in Figure 5.10 for $n = 3$. On this path, the concept names A_i, B_i act as a binary counter, with the first element after a on this path being labeled with $A_1, B_2, \dots, B_n$ and the 2^nth element on this path being labeled with $A_1, A_2, \dots, A_n$. The number of different concept name labels on this path is 2^n, indicating that no model of polynomial size can be ELQ-universal.

As an additional obstacle, the fitting problem for ELIQs under $\mathcal{ELI}$ ontologies is known to be undecidable [Fun+19].

These properties of $\mathcal{ELI}$ indicate that we cannot expect the learning approach for $\mathcal{EL}^r$ ontologies to work for $\mathcal{ELI}$ ontologies. Indeed, already the essential $\mathsf{minimize}_{\mathcal{O}}$ subroutine does not work in polynomial time under $\mathcal{ELI}$ ontologies. Hence, we can only hope to achieve polynomial query learning under $\mathcal{ELI}$ ontologies. However, we show that even permitting CQ-equivalence queries does not suffice to enable polynomial query learning under $\mathcal{ELI}$ ontologies. This already holds for the ELQs. Like the lower bound proofs in Chapter 4, we show this using an Angluin-style argument, arguing that the learner cannot obtain enough information to reliably identify the target query.

Theorem 5.50. *Every class of CQs that contains all ELQs is not polynomial query learnable under $\mathcal{ELI}$ ontologies with membership queries and CQ-equivalence queries.*

Proof. We use $\mathcal{ELI}$ ontologies $\mathcal{O}_n$ for $n \geq 1$, containing the following concept inclusions:

$$
\begin{array}{rll}
\top \sqsubseteq \exists r.\top \sqcap \exists s.\top & & \\
L_i \sqsubseteq \exists r.L_{i+1} \sqcap \exists s.L_{i+1} & & \text{for } 0 \leq i < n \\
L_i \sqsubseteq \exists r.L_{i+1} & & \text{for } n \leq i < 2n \\
L_{2n} \sqsubseteq A & & \\
\exists \sigma.L_{i+1} \sqsubseteq L_i & & \text{for } \sigma \in \{r,s\} \text{ and } 0 \leq i < 2n \\
K_i \sqsubseteq \exists \sigma.(K_{i+1} \sqcap V^{\sigma}_{i+1}) & & \text{for } \sigma \in \{r,s\} \text{ and } 0 \leq i < n \\
K_i \sqcap W^{\sigma}_{i-n+1} \sqsubseteq \exists r.K_{i+1} & & \text{for } \sigma \in \{r,s\} \text{ and } n \leq i < 2n \\
\exists \sigma^-.(K_j \sqcap V^{\sigma'}_i) \sqsubseteq V^{\sigma'}_i & & \text{for } \sigma,\sigma' \in \{r,s\},\ 1 \leq i \leq n, \\
 & & \text{and } i \leq j \leq 2n \\
K_{2n} \sqcap V^{\sigma}_i \sqcap W^{\overline{\sigma}}_i \sqsubseteq A & & \text{for } \sigma \in \{r,s\} \text{ and } 1 \leq i \leq n \\
\exists \sigma.W^{\sigma'}_i \sqsubseteq W^{\sigma'}_i & & \text{for } \sigma \in \{r,s,r^-,s^-\}, \\
 & & \sigma' \in \{r,s\}, \text{ and } 1 \leq i \leq n \\
W^r_i \sqcap W^s_i \sqsubseteq L_0 & & \text{for } 1 \leq i \leq n \\
\exists \sigma.K_{i+1} \sqsubseteq K_i & & \text{for } \sigma \in \{r,s\} \text{ and } 0 \leq i < 2n \\
\exists \sigma^-.\top \sqsubseteq U^{\sigma}_1 & & \text{for } \sigma \in \{r,s\} \\
\exists \sigma^-.U^{\sigma'}_i \sqsubseteq U^{\sigma'}_{i+1} & & \text{for } \sigma,\sigma' \in \{r,s\} \text{ and } 1 \leq i < 2n \\
U^r_i \sqcap U^s_i \sqsubseteq D & & \text{for } 1 \leq i \leq 2n \\
K_i \sqcap A \sqsubseteq D & & \text{for } 0 \leq i < 2n \\
L_i \sqcap A \sqsubseteq D & & \text{for } 0 \leq i < 2n \\
L_i \sqcap L_j \sqsubseteq D & & \text{for } n \leq i < j \leq 2n
\end{array}
$$

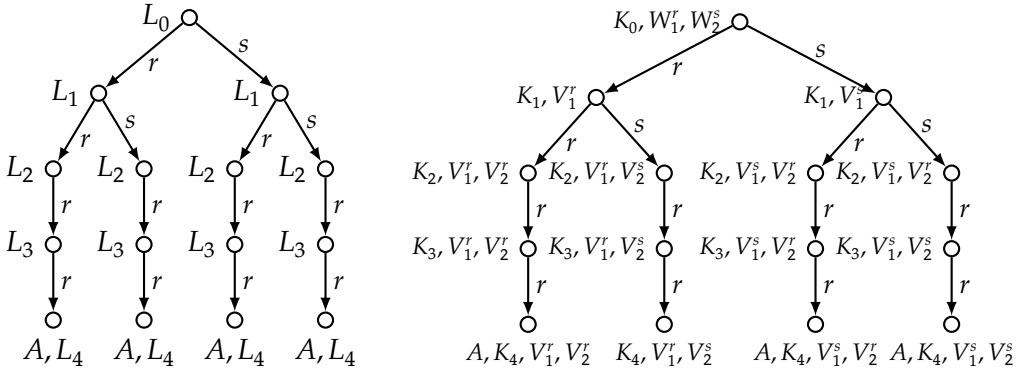

Figure 5.11: The interpretations L_0-tree (left) and K_0-tree (right) for $n = 2$ and $\mathcal{W} = \{1, 2\}$.

$$
\begin{aligned}
K_i \sqcap K_j &\sqsubseteq D && \text{for } n \leq i < j \leq 2n \\
L_i \sqcap K_j &\sqsubseteq D && \text{for } n \leq i, j \leq 2n \\
\exists \sigma.D &\sqsubseteq D && \text{for } \sigma \in \{r, s, r^-, s^-\} \\
D &\sqsubseteq L_0
\end{aligned}
$$

where $\overline{r} = s$ and $\overline{s} = r$. The size of the used signature Σ_n is polynomial in n. Every $\mathcal{O}_n$ is associated with a set H_n of 2^n potential target queries of the form

$$q(x_0) \leftarrow \sigma_1(x_0, x_1) \wedge \cdots \wedge \sigma_n(x_{n-1}, x_n) \wedge r(x_n, x_{n+1}) \wedge \cdots \wedge r(x_{2n-1}, x_{2n}) \wedge A(x_{2n})$$

with $\sigma_1, \ldots, \sigma_n \in \{r, s\}$.

Assume to the contrary of what is to be shown that ELQs are polynomial query learnable under $\mathcal{ELI}$ ontologies when unrestricted CQs can be used in equivalence queries. Then, there exists a learning algorithm and a polynomial p such that at any time, the sum of the sizes of the inputs to membership and equivalence queries made so far is bounded by $p(n_\Sigma, n_\mathcal{O}, n_{q_T}, n_\mathcal{A})$, where n_Σ is the size of the used signature, $n_\mathcal{O}$ the size of the used ontology, n_{q_T} the size of the target query and $n_\mathcal{A}$ the size of the largest counterexample seen so far.

We choose n such that $2^n > p(f_1(n), f_2(n), f_3(n), f_3(n)) + 1$ where f_1, f_2, f_3 are polynomials such that for every $n \geq 1$, $|\Sigma_n| \leq f_1(n)$, $\|\mathcal{O}_n\| \leq f_2(n)$, $\|q\| \leq f_3(n)$ for all $q \in H_n$ and f_4 bounds from above the size of all counterexamples returned by the teacher that we craft below. Consider then $\mathcal{O}_n$ and H_n as defined above. We let the teacher maintain a set of hypotheses H, starting with $H = H_n$ and then proceeding to subsets thereof, such that at no point the learner can distinguish between any of the candidate targets in H.

More precisely, consider a membership query with the data example $(\mathcal{A}, a_0)$. The teacher responds as follows:

1. if $\mathcal{A}, \mathcal{O}_n \models L_0(a_0)$, then answer *yes*;
2. if $\mathcal{A}, \mathcal{O}_n \models K_0(a_0)$ and there are $\sigma_1, \dots, \sigma_n \in \{r, s\}$ with $\mathcal{A}, \mathcal{O}_n \models W_i^{\sigma_i}(a_0)$ for $1 \leq i \leq n$, then answer *yes* and remove the ELQ

   $$q(x_0) \leftarrow \sigma_1(x_0, x_1) \wedge \cdots \wedge \sigma_n(x_{n-1}, x_n) \wedge r(x_n, x_{n+1}) \wedge \cdots \wedge r(x_{2n-1}, x_{2n}) \wedge A(x_{2n})$$

   from H, if present;
3. otherwise, answer no and remove all q with $\mathcal{A}, \mathcal{O}_n \models q(a_0)$ from H.

Higher up rules have higher priority, for example, Case 2 is applied only if Case 1 does not apply. To understand this strategy, consider the consequences of the concept name L_0 and the concept name K_0 under $\mathcal{O}_n$ displayed in Figure 5.11 for $n = 2$. We formally describe them later. It is not difficult to verify that the answers are correct regarding the hypothesis set H that remains after the answer is given.

Now consider an equivalence query with CQ $q_H(x_0)$. The teacher responds as follows:

1. if $\{L_0(a_0)\}, \mathcal{O}_n \not\models q_H(a_0)$, then return $\{L_0(a_0)\}$ as positive counterexample;
2. if $\{\top(a_0), L_0(a_1)\}, \mathcal{O}_n \models q_H(a_0)$, then return $\{\top(a_0), L_0(a_1)\}$ as a negative counterexample;
3. if there are $\sigma_1, \dots, \sigma_n \in \{r, s\}$ such that

   $$\{K_0(a_0), W_1^{\sigma_1}(a_0), \dots, W_n^{\sigma_n}(a_0)\}, \mathcal{O}_n \not\models q_H(a_0),$$

   then choose such $\sigma_1, \dots, \sigma_n$, return $\{K_0(a_0), W_1^{\sigma_1}(a_0), \dots, W_n^{\sigma_n}(a_0)\}$ as a positive counterexample, and remove the ELQ

   $$q(x_0) \leftarrow \sigma_1(x_0, x_1) \wedge \cdots \wedge \sigma_n(x_{n-1}, x_n) \wedge r(x_n, x_{n+1}) \wedge \cdots \wedge r(x_{2n-1}, x_{2n}) \wedge A(x_{2n})$$

   from H (if present).

Again, higher up rules have higher priority and the answers are always correct with respect to the updated set H. For Case 3, note that the counterexample $\mathcal{A} = \{K_0(a_0), W_1^{\sigma_1}(a_0), \dots, W_n^{\sigma_n}(a_0)\}$ is such that $\mathcal{A}, \mathcal{O}_n \models q(a_0)$ for all ELQs

$$q(x_0) \leftarrow \sigma'_1(x_0, x_1) \wedge \cdots \wedge \sigma'_n(x_{n-1}, x_n) \wedge r(x_n, x_{n+1}) \wedge \dots \wedge r(x_{2n-1}, x_{2n}) \wedge A(x_{2n})$$

with $\sigma'_1, \dots, \sigma'_n \in \{r, s\}$ except $\sigma'_1 \cdots \sigma'_n = \sigma_1 \cdots \sigma_n$.

We argue that Cases 1 to 3 are exhaustive. If $\mathcal{A}_{q_H}, \mathcal{O}_n \models L_0(x_0)$, then

$$\{K_0(a_0), W_1^{\sigma_1}(a_0), \dots, W_n^{\sigma_n}(a_0)\}, \mathcal{O}_n \not\models q_H(a_0)$$

for any $\sigma_1, \dots, \sigma_n, \in \{r, s\}$ and thus Case 3 is applicable.

For the case $\mathcal{A}_{q_H}, \mathcal{O}_n \not\models L_0(x_0)$, assume that Cases 1 and 2 do not apply. Let q'_H be the restriction of q_H to variables that are reachable from x_0 and let q''_H be the restriction of q_H to the variables that are not reachable.

Non-applicability of Case 1 implies that there is a homomorphism h from q_H to $\mathcal{U}_{\{L_0(a_0)\},\mathcal{O}_n}$ with $h(x_0) = a_0$. Consequently, q_H can only contain the symbols r, s, A as well as the concept names L_i and U_i^σ.

If there is no variable $x \in \mathsf{var}(q'_H)$ such that $A(x) \in q'_H$, then h is also a homomorphism from q'_H to $\mathcal{U}_{\{\top(a_0)\},\mathcal{O}_n}$ due to the first concept inclusion in $\mathcal{O}_n$. Taking the union of h with a homomorphism from q''_H to $\mathcal{U}_{\{L_0(a_1)\},\mathcal{O}_n}$ yields a homomorphism g from q_H to $\mathcal{U}_{\{\top(a_0),L_0(a_1)\},\mathcal{O}_n}$ with $g(x_0) = a_0$. This contradicts that Case 2 does not apply. Therefore, there must be a variable $x \in \mathsf{var}(q'_H)$ such that $A(x) \in q_H$. Since $\{L_0(a_0)\}, \mathcal{O}_n \models q_H(x_0)$, x must be reachable from x_0 on an r/s-path of length exactly $2n$ whose last n components are all r. Let the first n components be $\sigma_1, \dots, \sigma_n$. Then

$$\{K_0(a_0), W_1^{\sigma_1}(a_0), \dots, W_n^{\sigma_n}(a_0)\}, \mathcal{O}_n \not\models q_H(a_0)$$

and thus Case 3 applies.

We next show the following, which is the most important consequence of the design of $\mathcal{O}_n$.

Claim. If $(\mathcal{A}, a_0)$ is given as a membership query and Cases 1 and 2 of membership queries do not apply, then

$$\|\mathcal{A}\| \geq |\{q \in H_n \mid \mathcal{A}, \mathcal{O}_n \models q(a_0)\}|.$$

Proof of the Claim. We may assume without loss of generality that $\mathcal{A}$ is connected. Since Cases 1 and 2 of membership queries do not apply, we observe the following properties:

(a) $\mathcal{A}, \mathcal{O}_n \not\models L_0(a_0)$.

(b) there are no $\sigma_1, \dots, \sigma_n \in \{r, s\}$ such that $\mathcal{A}, \mathcal{O}_n \models K_0(a_0)$ and $\mathcal{A}, \mathcal{O}_n \models W_i^{\sigma_i}(a_0)$ for $1 \leq i \leq n$.

By construction of $\mathcal{O}_n$, these properties imply the following properties for all i with $0 \leq i \leq 2n$:

(c) $\mathcal{A}$ contains no r/s-path of length i from a_0 to some a with $\mathcal{A}, \mathcal{O}_n \models L_i(a)$. In fact, the existence of such a path implies $\mathcal{A}, \mathcal{O}_n \models L_0(a_0)$, contradicting (a).

(d) $\mathcal{A}$ contains no r/s-path of length i from a_0 to some a with $\mathcal{A}, \mathcal{O}_n \models K_i(a)$ and assertions $W_1^{\sigma_1}(a_1), \dots, W_n^{\sigma_n}(a_n)$ with $\sigma_1, \dots, \sigma_n \in \{r, s\}$. In fact, the existence of such a path and such assertions implies $\mathcal{A}, \mathcal{O}_n \models K_0(a_0)$ and $\mathcal{A}, \mathcal{O}_n \models W_i^{\sigma_i}(a_0)$ for $1 \leq i \leq n$, contradicting (b).

(e) $\mathcal{A}$ contains no r/s-paths p_1, p_2 of length i that end at the same individual and such that p_1 starts with an r-edge while p_2 starts with an s-edge. In fact, the existence of such paths and the connectedness of $\mathcal{A}$ implies $\mathcal{A}, \mathcal{O}_n \models D(a_0)$, and thus $\mathcal{A}, \mathcal{O}_n \models L_0(a_0)$, contradicting (a).

We now sketch the construction of a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{O}_n$. Let

$$\mathcal{W} = \{i \in \{1, \dots, n\} \mid W_i^\sigma(a) \in \mathcal{A}, \sigma \in \{r, s\}, a \in \mathsf{ind}(\mathcal{A})\}.$$

The following interpretations are used as building blocks for $\mathcal{I}$:

- an L_i*-path*, for $n \leq i \leq 2n$, is an r-path of length $2n - i$ that makes L_{i+j} true at the node at distance $j \in \{0, \dots, n - i\}$ from the start of the path and that makes true A at the end of the path;
- a K_i*-path*, for $n \leq i \leq 2n$, is defined as follows; let ℓ be maximal such that $\{(i - n) + 0, \dots, (i - n) + \ell\} \subseteq \mathcal{W}$; then a K_i-path is an r-path of length ℓ that makes K_{i+j} true at the node at distance $j \in \{0, \dots, \ell\}$ from the start of the path and that makes true A at the node at distance $2n - i$ (if it exists); in addition, the start of the path might make true any of the concept names V_j^σ, $\sigma \in \{r, s\}$ and $1 \leq j \leq n$, which are then all also made true by all other nodes on the path;
- an L_i*-tree*, for $0 \leq i < n$, is a binary r/s-tree of depth $n - i$ that makes L_{i+j} true at every node on level $j \in \{0, \dots, n - i\}$; in addition, every node on level $n - i$ is the start of an L_n-path;
- a K_i*-tree*, for $0 \leq i < n$, is a binary r/s-tree of depth $n - i$ that makes true K_{i+j} at every node on level $j \in \{0, \dots, n - 1\}$ and V_{i+j}^σ at every node on level at least j that is a σ-successor of its parent; in addition, every node on level $n - i$ is the start of a K_n-path; moreover, the root might make true any of the concept names V_j^σ, $\sigma \in \{r, s\}$ and $1 \leq j \leq i$, which are then all also made true by all other nodes in the tree.

In all of the above, any node that has an incoming r/s-path of length $i \in \{1, \dots, 2n\}$ that starts with $\sigma \in \{r, s\}$ is additionally labeled with the concept name U_i^σ. Moreover, the beginning of the path/root of the tree may be labeled with concept names of the form U_i^σ. Then, any node on depth $i + j$ with $i + j \leq 2n$ is labeled with U_{i+j}^σ.

Figure 5.11 visualizes these building blocks The concept names U_i^σ and W_i^σ are left out and mentioned just once, respectively. Note that the K_0-tree is missing a concept name A at the K_2-path specified by W_1^r, W_2^s.

Now, we construct $\mathcal{I}$ by starting with $\mathcal{A}$ and doing the following:

1. exhaustively apply all concept inclusions in $\mathcal{O}_n$ that have a concept name on the right-hand side;
2. if $a \in L_i^\mathcal{I}$, $0 \leq i < n$, then attach at a an L_i-tree;
3. if $a \in K_i^\mathcal{I}$, $0 \leq i < n$, then attach at a a K_i-tree;
4. if $a \in L_i^\mathcal{I}$, $n \leq i \leq 2n$, then attach at a an L_i-path;
5. if $a \in K_i^\mathcal{I}$, $n \leq i \leq 2n$, then attach at a a K_i-path;
6. at every $a \in \Delta^\mathcal{I}$, attach an infinite tree in which every node has two successors, one for each role name r, s, and in which no concept names are made true;
7. if $W_i^\sigma(a) \in \mathcal{A}$ for some a, then make W_i^σ true everywhere in $\mathcal{I}$.

By going over the concept inclusions in $\mathcal{O}_n$ and using Properties (a) and (b), it can be verified that $\mathcal{I}$ is indeed a model of $\mathcal{O}_n$. In particular, the inclusions $W_i^r \sqcap W_i^s \sqsubseteq L_0$ are satisfied since there is no $d \in \Delta^\mathcal{I}$ with $d \in (W_i^r \sqcap W_i^s)$; if there was such a d, then by construction of $\mathcal{I}$ there would be assertions $W_i^r(a)$ and $W_i^s(b)$ in $\mathcal{A}$, in contradiction to the connectedness of $\mathcal{A}$ and $\mathcal{A}, \mathcal{O}_n \not\models L_0(a_0)$. For the concept inclusions $U_i^r \sqcap U_i^s \sqsubseteq L_0$, we argue that there is no $d \in \Delta^\mathcal{I}$ with $d \in (U_i^r \sqcap U_i^s)^\mathcal{I}$. To see this, note that there are no $U_i^r(a), U_i^s(a) \in \mathcal{A}$ for any a as otherwise $\mathcal{A}, \mathcal{O}_n \models L_0(a_0)$. Now consider Step 1 of the construction of $\mathcal{I}$ and assume that it adds some $a \in \mathsf{ind}(\mathcal{A})$ to $(U_i^r \sqcap U_i^s)^\mathcal{I}$. But this means that $\mathcal{A}, \mathcal{O}_n \models U_i^r(a)$ and $\mathcal{A}, \mathcal{O}_n \models U_i^s(a)$, in contradiction to $\mathcal{A}, \mathcal{O}_n \not\models L_0(a_0)$, due to connectedness of $\mathcal{A}$. Given that there is no $d \in (U_i^r \sqcap U_i^s)^\mathcal{I}$ for any i after Step 1, it is readily checked that the elements d added in Steps 2–6 also satisfy $d \notin (U_i^r \sqcap U_i^s)^\mathcal{I}$.

We now use $\mathcal{I}$ to prove the claim. Let H' be the set of all $q \in H_n$ with $\mathcal{A}, \mathcal{O}_n \models q(a_0)$. With each $q \in H'$, we associate an $a_q \in \mathsf{ind}(\mathcal{A})$ as follows. Let q be the CQ $q(x_0) \leftarrow \sigma_1(x_0, x_1) \wedge \cdots \wedge \sigma_{2_n}(x_{2n-1}, x_{2n}) \wedge A(x_{2n})$. Then $\mathcal{I}$ contains a path from a_0 to some element $d_q \in A^\mathcal{I}$ that is labeled $\sigma_1 \cdots \sigma_{2n}$. If $d_q \in \mathsf{ind}(\mathcal{A})$, then $a_q = d_q$. Otherwise, d_q is in a path or tree attached to some $a \in \mathsf{ind}(\mathcal{A})$. Set $a_q = a$. To show that $\|\mathcal{A}\| \geq |\{q \in H_n \mid \mathcal{A}, \mathcal{O}_n \models q(a_0)\}|$ as required, it suffices to prove that $a_q \neq a_{q'}$ whenever $q \neq q'$. Thus, let $q, q' \in H'$ with $q \neq q'$,

$$q(x_0) \leftarrow \sigma_1(x_0, x_1) \wedge \cdots \wedge \sigma_{2n}(x_{2n-1}, x_{2n}) \wedge A(x_{2n}),$$
$$q'(x_0') \leftarrow \sigma_1'(x_0', x_1') \wedge \cdots \wedge \sigma_{2n}'(x_{2n-1}', x_{2n}') \wedge A(x_{2n}').$$

Assume to the contrary of what is to be shown that $a_q = a_{q'}$. We distinguish four cases:

- $d_q = a_q, d_{q'} = a_{q'}$.

  Then there is a path from a_0 to a_q in $\mathcal{I}$ labeled $\sigma_1 \cdots \sigma_{2n}$ and a path from a_0 to $a_{q'}$ labeled $\sigma'_1 \cdots \sigma'_{2n}$. By construction of $\mathcal{I}$, these paths must already exist in $\mathcal{A}$. From $q \neq q'$, we thus obtain a contradiction to Property (e).

- $d_q = a_q, d_{q'} \neq a_{q'}$.

  By construction of $\mathcal{I}$, $d_{q'} \neq a_{q'}$ implies that $\mathcal{A}, \mathcal{O}_n \models L_i(a_q)$ or $\mathcal{A}, \mathcal{O}_n \models K_i(a_q)$ for some i with $0 \leq i \leq 2n$. Moreover, $d_q = a_q$ implies $\mathcal{A}, \mathcal{O}_n \models A(a_q)$. Thus, $\mathcal{A}, \mathcal{O}_n \models D(a_q)$. By the connectedness of $\mathcal{A}$, we obtain $\mathcal{A}, \mathcal{O}_n \models D(a_0)$, thus $\mathcal{A}, \mathcal{O}_n \models L_0(a_0)$ in contradiction to Property (c).

- $d_q \neq a_q, d_{q'} = a_{q'}$.

  Symmetric to previous case.

- $d_q \neq a_q, d_{q'} \neq a_{q'}$.

  We first show that d_q and $d_{q'}$ are not in an L_i-tree, for $0 \leq i < n$. Assume to the contrary that d_q is (the case of $d_{q'}$ is symmetric). Then it occurs on level $2n - i$ in the tree, since $d_q \in A^{\mathcal{I}}$. Since an L_i-tree was attached to a_q, we must have $\mathcal{A}, \mathcal{O}_n \models L_i(a_q)$. Moreover, there is an r/s-path in $\mathcal{I}$ from a_0 to a_q of length i, the prefix of $\sigma_1 \cdots \sigma_{2n}$ of this length. By construction of $\mathcal{I}$, this path must already be in $\mathcal{A}$. This is in contradiction to Property (c).

  We next show that d_q and $d_{q'}$ are not in a K_i-tree, $0 \leq i < n$. Assume to the contrary that d_q is (the case of $d_{q'}$ is symmetric). Then it occurs on level $2n-i$ in the tree, since $d_q \in A^{\mathcal{I}}$. By definition of such trees (and the attached paths), this implies that $\mathcal{W} = \{1, \dots, n\}$ and thus $\mathcal{A}$ contains assertions $W_1^{\sigma''_1}(a_1), \dots, W_n^{\sigma''_n}(a_n)$. We must further have $\mathcal{A}, \mathcal{O}_n \models K_i(a_q)$ and there is an r/s-path in $\mathcal{I}$, thus in $\mathcal{A}$ from a_0 to a_q of length i. This is in contradiction to Property (d).

  Thus, d_q and d'_q are both in an L_i-path or in a K_i-path, $n \leq i \leq 2n$. If they are in different such paths, then $\mathcal{A}, \mathcal{O}_n \models D(a_q)$, which is in contradiction to Property (c) as $\mathcal{A}$ is connected. Thus, they must be in the same L_i-path or in the same K_i-path. Since each such path contains a single element d with $d \in A^{\mathcal{I}}$, we obtain $d_q = d'_q$. From $q \neq q'$, it thus follows that there are two different paths of length i in $\mathcal{I}$ from a_0 to a_q, the prefixes of this length of $\sigma_1 \cdots \sigma_{2n}$ and of $\sigma'_1 \cdots \sigma'_{2n}$. This is in contradiction to Property (e).

This finishes the proof of the claim.

We can use the claim to show the following invariant:

(∗) at every point, the sum m of the sizes of the inputs to membership and equivalence queries made so far is not smaller than the number of candidates that were removed from H.

Note that the teacher removes candidate queries from H only in Cases 2 and 3 of membership queries (Page 156) and Cases 3 and 6 of equivalence queries (Page 156). In all cases except Case 3 of membership queries, only one candidate is removed from H. The claim implies that the number of removed candidates in Case 3 of membership queries is bounded from above by the size of the query posed.

It then follows that there is a polynomial f_4 such that the size of all counterexamples returned by the oracle is at most $f_4(n)$. The overall sum of the sizes of posed membership and equivalence queries is bounded by $p(f_1(n), f_2(n), f_3(n), f_4(n))$. Hence, we can conclude from (∗) that at most $p(f_1(n), f_2(n), f_3(n), f_4(n))$ candidate concepts have been removed from H_n. By the choice of n, at least two candidate concepts remain in H after the algorithm finishes. Thus, the learner cannot distinguish between them, and we have derived a contradiction. □

Theorem 5.50 also precludes the possibility of polynomial time learning under $\mathcal{ELI}$ ontologies with an $\mathcal{ELI}$ reasoning oracle.

5.6 Queries with Disjunctions

All query classes we have considered so far are subclasses of CQs and can therefore only express conjunction of atoms. In many practical scenarios, it is also desirable to write queries using *disjunctions*, for example when querying a database for all things that are a dog or a cat. In this chapter, we briefly consider the learnability of queries that use disjunctions. One way to use disjunctions in queries are *unions of conjunctive queries* (UCQs). A UCQ is an expression of the form

$$q(\overline{x}) \leftarrow q_1(\overline{x}) \vee \cdots \vee q_n(\overline{x})$$

where $q_1, \dots, q_n$ are CQs of the same arity. A tuple $\overline{d}$ is an *answer to a UCQ* q in an interpretation $\mathcal{I}$, written $\mathcal{I} \vDash q(\overline{d})$ if there is an i with $1 \leq i \leq n$ such that $\mathcal{I} \vDash q_i(\overline{d})$. This allows us to express disjunctions like

$$q(x) \leftarrow \mathsf{Dog}(x) \vee \mathsf{Cat}(x).$$

Ten Cate, Dalmau, and Kolaitis show that UCQs are polynomial time learnable using membership and equivalence queries [tCDK13] under the empty ontology,

in fact, using a variant of Algorithm 5.1. Therefore, the question arises whether our result concerning the polynomial time learnability of subclasses of CQs under ontologies can be transferred to UCQs. We conjecture that this is the case and that Algorithm 5.2 and Algorithm 5.3 can be extended analogously to learn *unions of* ELIQs, or *unions of* CQ^{csf} queries, respectively. This new algorithm would maintain a union of CQ^{csf} queries as a hypothesis, and would use counterexamples and membership queries to update single CQ^{csf} queries in this union, or to add new elements to the union. We leave a proof of this for future work.

A known property of UCQs is that, since disjunctions appear nested below conjunctions, some queries cannot be represented succinctly. For example, if we want to query for all things that are (A_1 or A_2) and (A_3 or A_4), we can only formulate this as the UCQ

$$q(x) \leftarrow (A_1(x) \land A_3(x)) \lor (A_1(x) \land A_4(x)) \lor (A_2(x) \land A_3(x)) \lor (A_2(x) \land A_4(x)).$$

UCQs can therefore be exponentially larger than queries written in a query language that can freely nest disjunctions and conjunctions.

One way to define such a query language is to use $\mathcal{ELU}$ concept queries, that is $\mathcal{EL}$ concepts extended with disjunctions. An $\mathcal{ELU}$ *concept* is formed according to the syntax rule

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

where A ranges over $\mathsf{N_C}$ and r over $\mathsf{N_R}$. The semantics of $\top$, $\bot$, $\sqcap$ and $\exists$ is defined exactly as in Section 3.1. We extend the interpretation function $\cdot^{\mathcal{I}}$ for $\sqcup$ by setting

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}.$$

$\mathcal{ELU}$ queries allow us to express the above query more succinctly as the concept

$$(A_1 \sqcup A_2) \sqcap (A_3 \sqcup A_4).$$

Unfortunately, this succinct representation makes learning $\mathcal{ELU}$ queries with membership queries and equivalence queries hard, which we show next. So far, in Section 4.1 and Section 5.5, we have used the basic combinatorial technique introduced by Angluin [Ang88b] to show lower bounds for exact learning. It is unclear if or how this technique can be applied to $\mathcal{ELU}$ queries. Instead, we will rely on the hardness of learning *monotone Boolean* (*propositional*) *formulas*, that is Boolean formulas that do not use negation, only conjunction and disjunction.

Learning of Boolean formulas is one of the major fields where the exact learning framework has been applied. Many fragments of Boolean formulas have been shown to be polynomial time learnable, like k-term DNFs [Ang88b] or Horn formulas [AFP92]. However, the class of all Boolean formula has been shown to not

be polynomial time learnable with equivalence and membership queries, as long as certain cryptographic assumptions hold [AK95]. These assumptions are that one of the following problems cannot be solved in polynomial time: (1) testing quadratic residues modulo a composite, (2) inverting RSA encryption, (3) factoring Blum integers. See [KV89] for more details and a description of these problems. These assumptions all imply that P ≠ NP.

This lower bound already holds for monotone Boolean formulas.

Theorem 5.51 ([Dal99])**.** *Monotone Boolean formulas are not polynomial time learnable with membership queries and equivalence queries under cryptographic assumptions.*

Since $\mathcal{ELU}$ queries, like monotone Boolean formulas, only use conjunction and disjunction (but are unable to express Boolean negation), this directly allows us to show that $\mathcal{ELU}$ queries are not polynomial time learnable even in the case without an ontology via a trivial direct reduction.

Theorem 5.52. *$\mathcal{ELU}$ queries are not polynomial time learnable under the cryptographic assumptions.*

Proof. Assume to the contrary that there is a learning algorithm **A** that can learn $\mathcal{ELU}$ queries under the empty ontology in polynomial time using equivalence and membership queries.

We use **A** to formulate a polynomial time learning algorithm **A′** for monotone Boolean formulas. The new algorithm proceeds as follows.

- When started to learn a monotone Boolean formula over the propositional variables $x_1, \dots, x_n$, **A′** in turn starts **A** with the signature $\{A_1, \dots, A_n\}$, where $A_1, \dots, A_n$ are concept names.
- When **A** poses a membership query with the example $(\mathcal{A}, a)$, **A′** constructs the variable assignment $V_{\mathcal{A}}$ with $V_{\mathcal{A}}(x_i) = 1$ if and only if $\mathcal{A}, \varnothing \models A_i(a)$ for all i with $1 \leq i \leq n$, and poses a membership query with $V_{\mathcal{A}}$. **A′** returns the result of this membership query to **A**.
- When **A** poses an equivalence query with the $\mathcal{ELU}$ query q_H, **A′** computes the monotone Boolean formula φ_H using the following inductive translation f of $\mathcal{ELU}$ concepts:

$$\begin{aligned} f(A_i) &= x_i \\ f(C \sqcap D) &= f(C) \wedge f(D) \\ f(C \sqcup D) &= f(C) \vee f(D) \end{aligned}$$

  and setting $\varphi_H = f(q_H)$. The algorithm **A′** then poses an equivalence query with φ_H. If **A′** receives a valuation V as counterexample, it constructs an ABox $\mathcal{A}_V = \{A_i(a) \mid V(x_i) = 1\}$ and returns $(\mathcal{A}_V, a)$ as counterexample to **A**.

- When **A** terminates and returns an $\mathcal{ELU}$ query q_H, it returns $f(q_H)$ using the same translation as above.

Since $V \models \varphi$ if and only if $\mathcal{A}_V, \varnothing \models f(\varphi)$, the answers given to membership queries and equivalence queries are correct. Hence, $\mathbf{A}'$ returns in polynomial time a monotone Boolean formula that is equivalent to the target formula, contradicting Theorem 5.51. □

This observation justifies our focus on learning *conjunctive* queries. It is an interesting question, whether other results on the polynomial time learnability of classes of Boolean formulas can give rise to larger polynomial time learnable query classes.

5.7 Discussion

In this chapter, we have investigated the learnability of queries under ontologies using both membership queries and equivalence queries. Unsurprisingly, the addition of equivalence queries makes learning algorithms more powerful compared to the algorithms we considered in Chapter 4. The result can be summarized as follows.

- ELIQs are polynomial time learnable under *DL-Lite*$^{\mathcal{F}-}_{\text{horn}}$ (Theorem 5.17);
- ELQ, ELIQ$^{\text{sf}}$, and CQ$^{\text{csf}}$ queries are polynomial time learnable under $\mathcal{EL}^r$ ontologies if their arity is fixed (Theorem 5.39) and polynomial time learnable for unbounded arity if CQ-equivalence queries are permitted (Theorem 5.48);
- ELQ, ELIQ, and CQ queries are not polynomial query learnable under $\mathcal{ELI}$ ontologies, even if CQ-equivalence queries are permitted (Theorem 5.50);
- $\mathcal{ELU}$ queries are not polynomial time learnable under cryptographic assumption (Theorem 5.52).

Recall that the results about ELQs, ELIQs and ELIQ$^{\text{sf}}$ queries also apply to $\mathcal{EL}$ concepts, $\mathcal{ELI}$ concepts and symmetry-free $\mathcal{ELI}$ concepts, respectively.

The main issue we discussed in this chapter is how learning algorithms can update hypotheses with counterexamples obtained from equivalence queries to approach the target query. In Section 5.2 we used guided generalizations for this and products of compact models in Section 5.3. Combined with subroutines to minimize queries and to extract queries from the desired query class, this allows the learning algorithms to approach the target query in polynomially many steps. Next, we discuss how the results in this chapter might be generalized.

Using Counterexamples. Upon inspection of the conditions of generalization sequences (Definition 4.33) and the proof that the hypotheses of the learning algorithm form a generalization sequence, we can note that the only requirement for the new hypothesis q'_H is that $q_H \subseteq_{\mathcal{O}} q'_H \subseteq_{\mathcal{O}} q_T$ and $q'_H \not\subseteq_{\mathcal{O}} q_H$. In other words, it suffices that q'_H is an element of the frontier of q_H. In some sense, the LGGs (Definition 5.6), guided generalizations (Definition 5.9), and the product of compact models used in Section 5.3 enforce more properties than are necessary for the learning algorithm to make progress towards q_T. Of course, as noted in Chapter 4, the frontier of a hypothesis q_H under ontologies is often infinite or of exponential size and thus cannot be enumerated to search for q'_H. However, the counterexample returned by an equivalence query can provide us with more information, and may allow us to restrict the search to a small portion of the frontier. A more principled way to obtain learning algorithms could thus be based on *counterexample-guided frontiers*, that may be of polynomial size, in cases where the general frontier is prohibitively large.

Learning under $\mathcal{EL}_{\text{lhs}}$. As discussed in Section 5.1, ideally one would want to use the product $\mathcal{U}_{q_H,\mathcal{O}} \times \mathcal{U}_{\mathcal{A},\mathcal{O}}$ to update a hypothesis with a counterexample. This is of course not possible, if the universal models are infinite or of exponential size. For $\mathcal{EL}_{\text{lhs}}$ ontologies, that is ontologies where all concept inclusions have the shape $C \sqsubseteq A$ where C is an $\mathcal{EL}$ concept and A a concept name, it is always the case that $\mathcal{U}_{q,\mathcal{O}}$ is of polynomial size. By slightly modifying Algorithm 5.1, we can therefore easily show that CQs are polynomial time learnable under $\mathcal{EL}_{\text{lhs}}$ ontologies.

Functionality Constraints. Reasoning in $\mathcal{ELF}$, that is $\mathcal{EL}$ extended with functionality constraints, is ExpTime-complete [BBL05]. Hence, polynomial time learning of, for example, ELQs under $\mathcal{ELF}$ ontologies seems out of reach. Additionally, there are no ELQ-universal models of $\mathcal{ELF}$ ontologies that are of polynomial size, as demonstrated by Example 5.49, which can be expressed using $\mathcal{ELF}$ ontologies. We conjecture that the $\mathcal{ELI}$ ontologies used in the proof of Theorem 5.50 can be formulated in $\mathcal{ELF}$, and be used to show that ELQs are not polynomial query learnable under $\mathcal{ELF}$ ontologies. To show this, note that the roles s and r are functional in all used interpretations, and therefore each concept inclusion of the form $\exists \sigma^-.A \sqsubseteq B$, can be expressed in $\mathcal{ELF}$ as

$$A \sqsubseteq \exists \sigma.B, \quad \mathsf{func}(\sigma).$$

Role Inclusions. $\mathcal{ELH}$ ontologies, that is $\mathcal{EL}$ ontologies that additionally contain role inclusions, also possess ELQ-universal models of polynomial size. We can therefore use the same techniques as in Section 5.3 to show that ELQs are polynomial

time learnable under $\mathcal{ELH}$ ontologies. The situation becomes more difficult if we consider (symmetry-free) ELIQs, as the ELIQ $q(x_0) \leftarrow r(x_0, x_1) \wedge s(x_2, x_1)$ does not contain a symmetry, but still has spurious matches in compact models under the ontology $\mathcal{O} = \{s \sqsubseteq r\}$. Hence, a new definition of being symmetry-free is needed that takes into account the role inclusions in the ontology. We conjecture that sufficiently symmetry-free ELIQs and chordal CQs can also be learned in polynomial time under $\mathcal{ELH}$ ontologies.

CQs with Longer Cycles. The query class $\mathrm{CQ}^{\mathrm{csf}}$ allows only cycles of length at most 3 in a query before requiring a chord. It is an interesting question whether this can be generalized to longer cycles. The main issue is the construction of $C^3_{\mathcal{A},\mathcal{O}}$ and Lemma 5.24, which states that the anonymous part of $C^3_{\mathcal{A},\mathcal{O}}$ only contains cycles of length four or longer. It seems possible to construct compact models where every cycle has length at least k, and thus to allow cycles of length $< k$ in the query. One way to approach this is using *graphs of high girth* in the construction of a compact model, which are of polynomial size if k is fixed [Ott06], but their size in general depends exponentially on k.

Restricting Examples to Concepts. If we are interested in learning $\mathcal{EL}$ concepts, it might make sense to use concept examples instead of data examples, or equivalently to restrict data examples to be rooted and tree-shaped. In this setting, the learner aims to identify a target concept C_T. For an equivalence, query the learner produces a hypothesis concept C_H, and the teacher responds with a counterexample D such that $\mathcal{O} \models D \sqsubseteq C_H$ and $\mathcal{O} \not\models D \sqsubseteq C_T$ or $\mathcal{O} \not\models D \sqsubseteq C_H$ and $\mathcal{O} \models D \sqsubseteq C_T$. This restriction gives the teacher fewer options to choose counterexamples, and therefore the same ideas as in Section 5.1 to update a hypothesis with a counterexample still apply. For a membership query, the learner produces a concept D, and the teacher responds with *yes* if $\mathcal{O} \models D \sqsubseteq C_H$. As discussed in Section 4.7, this makes it impossible to use minimization as part of a subroutine like $\mathsf{extract}_Q$. Thus, it seems unlikely that a similar approach as the one in Algorithm 5.3 could be used to learn concepts under ontologies from concept examples.

What happens when q_T is not from Q? For practical scenarios, it is a natural question to consider how Algorithm 5.3 behaves if $q_T \notin Q$. It is easy to see that in this case the subroutine $\mathsf{extract}_Q$ must get stuck at some point, since the application of *Expand cycle* or *Split symmetry* will result in a query p with $p \not\subseteq_{\mathcal{O}} q_T$. This situation can easily be detected using membership queries. However, the point at which it gets stuck reveals only limited information about q_T. Algorithm 5.3 is therefore not useful to apply in this scenario.

Open Questions. Many interesting questions concerning learning with membership queries and equivalence queries remain open. The central ones are the following.

- Are (non-symmetry-free) ELIQs or CQs polynomial time learnable under $\mathcal{EL}^r$ ontologies? Are there other ways to update hypotheses with counterexamples than using compact models?
- Are CQs polynomial time learnable under *DL-Lite*$_{\text{horn}}$ or *DL-Lite*$_{\text{core}}$ ontologies?

Due to the difficulties discussed in this chapter, answering these questions positively or negatively will require new techniques and new constructions. If the answer to these questions are negative, then it is unclear how to approach it. The techniques we used to show Theorem 5.50 and Theorem 5.52 are candidates, but showing exact learning lower bounds for both query types is difficult.

In the next chapter, we consider the learnability of queries using only equivalence queries and the PAC learnability of queries.

Chapter 6

Learning from Examples

In Chapters 4 and 5, we have discussed the learnability of queries with only membership queries and with both membership queries and equivalence queries, respectively. It remains to answer if membership queries are really necessary for our results in Chapter 5 or if learning, for example, of ELQs under $\mathcal{EL}^r$ ontologies is also possible using only equivalence queries. We approach this question by considering PAC learnability of queries. Recall that, per Theorem 3.14, for query classes that fulfill a certain property, polynomial time learnability with equivalence queries implies polynomial time PAC learnability. Hence, if we show that polynomial time PAC learning of a certain query class that fulfills this property is not possible, then we can conclude that this query class cannot be learned in polynomial time with only equivalence queries.

An advantage of PAC learning algorithms over exact learning algorithms is that, in practical scenarios, a collection of labeled examples is easier to obtain than a capable teacher that can answer equivalence queries. In the second half of this chapter, we describe an implementation of a PAC learning algorithm, and perform experiments on real data.

Structure of This Chapter

We begin in Section 6.1 by showing that, if RP $\neq$ NP, there are no polynomial time PAC learning algorithms for many classes of CQs under the empty ontology. Most importantly, this applies to the classes of ELQ, ELIQs, and symmetry-free ELIQs, and thus complements the results in Chapter 5: both membership queries and equivalence queries are necessary to learn one of these query classes in polynomial time.

In Section 6.2 we thus turn away from polynomial time PAC learning, and instead consider *sample-efficient* PAC learning. We show that sample-efficient PAC learning algorithms exist for every query class and every ontology language, due to the bounded VC-dimension of query classes under ontologies.

Subsequently, we show in Section 6.3 that not every fitting algorithm is a sample-efficient PAC learning algorithm. Indeed, we show that algorithms that produce

one of three logically interesting kinds of fitting queries are not sample-efficient.

Then, in Section 6.4 we describe the implementation of one sample-efficient PAC learning algorithm for ELQs under $\mathcal{ELH}^r$ ontologies, that uses a SAT solver to compute fitting ELQs. We call this implementation *SAT-based PAC $\mathcal{EL}$ Concept Learner* or SPELL. We then compare SPELL to the existing ELQ learning system ELTL ($\mathcal{EL}$ tree learner) on various benchmarks in Section 6.5.

We conclude with discussing the results of this chapter in Section 6.6.

Related Publications

Section 6.1 is based on [tCat+24], but fixes a problem in the proof of Theorem 6.7. Section 6.3 presents results from the research note [tCat+23a]. Sections 6.2, 6.4 and 6.5 are based on [tCat+23c].

6.1 PAC Learning of Queries in Polynomial Time

We begin by showing that every class of CQs that contains all *path CQs* is not polynomial time PAC learnable. A path CQ is a unary CQ of the form

$$q(x_1) \leftarrow r(x_1, x_2) \wedge \cdots \wedge r(x_{n-1}, x_n) \wedge A(x_{j_1}) \wedge \cdots \wedge A(x_{j_m})$$

where r is a role name, A is a concept name and $1 \leq j_i \leq n$ for all $1 \leq i \leq m$.

We begin with reviewing the connection between PAC learning and the fitting problem. It is well known that in many learning settings, polynomial time PAC learning implies that the corresponding fitting problem can be solved in polynomial time, see for example [AB92, Theorem 6.2.1] or [PV88]. Many formulations of this connection do not state two implicit properties of their respective learning setting. Unfortunately, these properties do not necessarily hold in the setting of learning queries under ontologies. In what follows, we state this connection precisely.

For this, we restrict our attention to certain sets of labeled examples. If F is a set of data examples, then $F \times \{+, -\}$ is the set of all labeled data examples created from F, and $2^{F \times \{+,-\}}$ is the set of all sets of labeled data examples. For some F, let $M \subseteq 2^{F \times \{+,-\}}$ be a set of finite sets of labeled examples. The *fitting problem* for a query class $\mathcal{Q}$ and an ontology language $\mathcal{L}$ *on* M is defined similarly to the usual fitting problem (Definition 3.9) with the only modifiation being that the input set of labeled examples E must be from M.

Let $\mathcal{Q}$ be a query class, $\mathcal{L}$ an ontology language and M a set of finite sets of labeled examples. We say that $\mathcal{Q}$ has the *polynomial size fitting property* under $\mathcal{L}$ ontologies on M, if for every $\mathcal{L}$ ontology $\mathcal{O}$ and every $E \in M$, the existence of a fitting query

for E under $\mathcal{O}$ implies the existence of a fitting query whose size is bounded by a polynomial of $\|E\|$ and $\|\mathcal{O}\|$.

Recall that RP is the complexity class of all problems for which a probabilistic polynomial time Turing machine exists that answers *no* if the correct answer is *no*, and *yes* with probability ≥ 0.5 if the correct answer is *yes*. It is known that P $\subseteq$ RP $\subseteq$ NP and RP $\subseteq$ BPP $\subseteq$ P/poly. Since RP = NP thus implies that NP $\subseteq$ P/poly, which in turn implies via the Karp-Lipton theorem that the polynomial hierarchy collapses to the second level [AB09], it is generally believed that RP $\neq$ NP. Also note that RP $\neq$ NP implies that P $\neq$ NP.

We show the following.

Lemma 6.1. *Let $\mathbf{Q}$ be a query class, $\mathcal{L}$ an ontology language and M a set of finite sets of labeled examples such that $\mathbf{Q}$ queries can be answered in polynomial time under $\mathcal{L}$ ontologies on examples that occur in M, and $\mathbf{Q}$ has the polynomial size fitting property under $\mathcal{L}$ ontologies on M.*

If $\mathbf{Q}$ is polynomial time PAC learnable under $\mathcal{L}$, then the fitting problem for $\mathbf{Q}$ and $\mathcal{L}$ on M is in RP.

Proof. Assume that there is a (possibly randomized) polynomial time PAC learning algorithm $\mathbf{A}$ for $\mathbf{Q}$ and $\mathcal{L}$ with associated sample size m as in Definition 3.13. We use it to solve the fitting problem for $\mathbf{Q}$ and $\mathcal{L}$ in randomized polynomial time. Assume that a set $E \in M$ of k labeled examples, an $\mathcal{L}$ ontology $\mathcal{O}$ and a signature Σ are given as input. Let $n_{q_T} = p(\|E\|, \|\mathcal{O}\|)$, where p is the polynomial witnessing the fact that $\mathbf{Q}$ has the polynomial-size fitting property under $\mathcal{L}$. Let D be the distribution over E that assigns each example in E probability $1/k$, and let $n_{\mathcal{A}}$ be the maximum size of an example in E. Pick $\delta < 0.5$ and $\epsilon < 1/k$. We generate a new polynomial-sized collection of labeled examples E' by drawing $m(\frac{1}{\delta}, \frac{1}{\epsilon}, |\Sigma|, \|\mathcal{O}\|, n_{q_T}, n_{\mathcal{A}})$ samples from distribution D, and start $\mathbf{A}$ with inputs Σ, $\mathcal{O}$ and E'.

Since $\mathbf{A}$ is a polynomial time algorithm, there is a polynomial of $|\Sigma|$, $\|\mathcal{O}\|$, and $\|E'\|$ that bounds the running time of $\mathbf{A}$. Run $\mathbf{A}$ for that many steps. If $\mathbf{A}$ terminates within this time, check if its output fits E under $\mathcal{O}$. If so, we answer *yes*. Otherwise, if the output does not fit or if $\mathbf{A}$ does not terminate, we answer *no*.

This means that, if there is no fitting query, the output will be *no*. If, on the other hand, there is a fitting query, then there is one of size at most n, and hence, with probability $1 - \delta$, the algorithm $\mathbf{A}$ will output a query with error less than ϵ under $\mathcal{O}$ within the time bound. This, in fact, implies that the error is 0 because if the query misclassifies an example to which D assigns non-zero probability, then it will have error at least $1/k$. Hence, with probability $1 - \delta > 0.5$ the algorithm outputs *yes*.

Note that both polynomial time query answering and the polynomial fitting property are needed, in order for the described algorithm to run in polynomial time. □

Unfortunately, CQs, ELIQs, ELQs, and even path CQs do not have the polynomial fitting property on arbitrary M, as mentioned in Section 3.2.

Example 6.2. Consider examples $(\mathcal{A}_n, a_1)$ that consist of a directed cycle of length n where the last individual is labeled with A, that is

$$\mathcal{A}_n = \{r(a_1, a_2), \dots, r(a_n, a_1), A(a_n)\},$$

and the example $(\{r(b, b)\}, b)$ that is an unlabeled cycle of length 1. Let $p_1, \dots, p_m$ be numbers that are coprime. Then, the smallest path CQ that fits the examples

$$E = \{(\mathcal{A}_{p_i}, a_1, +) \mid 1 \leq i \leq m\} \cup \{(\{r(b, b)\}, b, -)\}$$

under the empty ontology has size at least $\prod_{1 \leq i \leq m} p_i$, but $\|E\|$ is polynomial in $\sum_{1 \leq i \leq m} p_i$.

Therefore, we need to employ a more involved argument to show that these classes are not polynomial time PAC learnable. We say that a CQ or ABox is *forest-shaped* if it is a disjoint union of tree-shaped CQs or ABoxes, respectively. Recall that tree-shaped means that the underlying graph forms a directed tree. To show non-polynomial time PAC learnability, we restrict the space of the examples we consider to only tree-shaped examples.

Lemma 6.3. *Let Σ be a signature that contains only one role name and any number of concept names. Given a unary CQ q over Σ,*

1. *we can test in polynomial time whether there exists a tree-shaped example $(\mathcal{A}, a)$ such that $\mathcal{A}, \emptyset \models q(a)$;*
2. *if the answer to the above question is positive, then we can construct in polynomial time a forest-shaped CQ q' over Σ such that for all tree-shaped examples $(\mathcal{A}, a)$, $\mathcal{A}, \emptyset \models q(a)$ if and only if $\mathcal{A}, \emptyset \models q'(a)$.*

Proof. It suffices to show that we can construct tree-shaped CQs from connected CQs. The general case then follows by component-wise analysis. Therefore, let q be a connected CQ. Additionally, let r be the role name in Σ.

Let $\sim$ be the smallest equivalence relation over the variables of q such that, whenever $r(u, v)$ and $r(u', v')$ are conjuncts of q and $v \sim v'$ then also $u \sim u'$. Let q' be the quotient of q with regard to $\sim$, that is, q' is obtained from q by choosing a representative of each $\sim$-equivalence class, and replacing every occurrence of a variable x by the representative of the $\sim$-equivalence class of x. It is easy to see that, for all tree-shaped examples $(\mathcal{A}, a)$, $\mathcal{A}, \emptyset \models q(a)$ if and only if $\mathcal{A}, \emptyset \models q'(a)$.

If q' contains a directed cycle, then $\mathcal{A}, \emptyset \not\models q'(a)$ for all tree-shaped examples $(\mathcal{A}, a)$. Hence, in this case, we are done.

If q' does not contain a directed cycle, there must exist a variable y for which q' does not contain any conjunct of the form $r(x, y)$. Furthermore, any simple path from y to any other variable z must consist entirely of forward edges. Otherwise, the path would be of the form

$$r(y, x_1), \dots, r(x_{i-1}, x_i), r^-(x_i, x_{i+1}), \dots, r(x_j, z)$$

and x_{i-1} and x_{i+1} would have been identified when we constructed q'. It follows that q' is tree-shaped. Furthermore, let $(\mathcal{A}_{q'}, a_{q'})$ be the canonical example of q'. Then, clearly, $\mathcal{A}_{q'}, \emptyset \models q'(a_{q'})$. □

Because of Lemma 6.3, we can evaluate unary (potentially cyclic) CQs that only use one role name over tree-shaped examples in polynomial time. Let $(\mathcal{A}, a)$ be a tree-shaped example and q a unary CQ. First, construct a forest-shaped unary CQ q' according to Lemma 6.3 in polynomial time and then decide whether $\mathcal{A}, \emptyset \models q'(a)$, which is possible in polynomial time [Yan81]. We therefore say that for every unary class of CQs $\mathcal{Q}$ over Σ, $\mathcal{Q}$ queries can be answered in polynomial time over tree-shaped examples. Note that his does not contradict that deciding the existence of homomorphisms to acyclic interpretations is still NP-complete [HNZ96], as the needed interpretations are not directed trees and thus are not tree-shaped according to our definition. We use this fact for showing that there is a class of tree-shaped examples for which the fitting problem is NP-hard and for which CQs have the polynomial fitting property.

To obtain these tree-shaped examples, we use a reduction from the satisfiability problem for 3CNF formulas, building on reductions given by Kietz [Kie93] and Haussler [Hau89]. Let $\varphi = \varphi_1 \wedge \dots \wedge \varphi_k$ be a 3CNF formula over the propositional variables $\{X_1, \dots, X_m\}$. We denote by $L = \{X_i, \neg X_i \mid 1 \leq i \leq m\}$ the set of all literals over $\{X_1, \dots, X_m\}$. With every literal $l \in L$, we associate a natural number j_l as follows. Set $j_l = 2i$ if l is of the form X_i and $j_l = 2i - 1$ if l is of the form $\neg X_i$. Now, define an ABox $\mathcal{A}_\varphi$ to contain the following assertions:

- $r(a_i, p_{i,1})$ and $r(a_i, n_{i,1})$ for $1 \leq i \leq m$,
- $r(p_{i,j}, p_{i,j+1})$ and $r(n_{i,j}, n_{i,j+1})$ for $1 \leq i \leq m$ and $j < 2m$,
- $A(p_{i,j_l})$ for every literal $l \in L \setminus \{\neg X_i\}$,
- $A(n_{i,j_l})$ for every literal $l \in L \setminus \{X_i\}$,
- $r(b, b_{i,1})$ for $1 \leq i \leq k$,
- $r(b_{i,j}, b_{i,j+1})$ for $1 \leq i \leq k$ and $j < 2m$,
- $A(b_{i,j_l})$ for every $l \in L$ and $i \leq k$ with l not occurring in the clause φ_i.

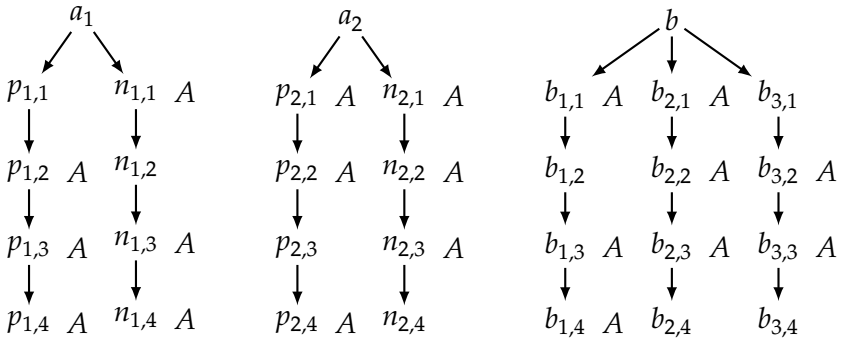

Figure 6.1: The ABox $\mathcal{A}_\varphi$ for the formula $\varphi = X_1 \wedge X_2 \wedge (\neg X_1 \vee X_2)$.

Then set

$$E_\varphi = \{(\mathcal{A}_\varphi, a_i, +) \mid 1 \leq i \leq m\} \cup \{(\mathcal{A}_\varphi, b, -)\}.$$

Example 6.4. Consider the 3CNF formula $\varphi = X_1 \wedge X_2 \wedge (\neg X_1 \vee X_2)$ over the propositional variables $\{X_1, \dots, X_m\}$. The ABox $\mathcal{A}_\varphi$ is displayed in Figure 6.1, where each edge represents an r assertion. Note that since φ is satisfiable, there is a path CQ that fits the labeled examples $E_\varphi = \{(\mathcal{A}_\varphi, a_1, +), (\mathcal{A}_\varphi, a_2, +), (\mathcal{A}_\varphi, b, -)\}$, namely

$$q(x_0) \leftarrow r(x_0, x_1) \wedge r(x_1, x_2) \wedge r(x_2, x_3) \wedge r(x_3, x_4) \wedge A(x_2) \wedge A(x_4).$$

Lemma 6.5. *For all 3CNF formulas φ over $\{X_1, \dots, X_m\}$:*

1. *From a satisfying assignment for φ, one can construct a path CQ that fits E_φ in polynomial time.*
2. *Conversely, if there is a CQ that fits E_φ, then φ has a satisfying assignment.*

In particular, whenever there is a CQ that fits E_φ, then there is a fitting path CQ of size polynomial in m.

Proof. We begin with Point 1. Let v be a satisfying assignment for φ. Let

$$q(x_0) \leftarrow r(x_0, x_1) \wedge \cdots \wedge r(x_{2m-1}, x_{2m}) \wedge \bigwedge_{l \in L \text{ such that } v \models l} A(x_{j_l}).$$

Using the construction of $\mathcal{A}_\varphi$ it can be verified that $\mathcal{A}_\varphi, \varnothing \models q(a_i)$ for $1 \leq i \leq m$ and $\mathcal{A}_\varphi, \varnothing \not\models q(b)$.

We continue with Point 2. Let $q(x)$ be a unary CQ that fits E_φ. By Lemma 6.3, we may assume that q is forest-shaped. Furthermore, we may assume without loss of generality that q is connected. If x has an r-predecessor in q, then q does not fit the positive examples of E_φ. Hence, x is the root of the tree in q.

Since $q(x)$ fits the negative example $(\mathcal{A}_\varphi, b)$, $\mathcal{A}_\varphi, \varnothing \not\models q(b)$. This means that either (i) q contains an atom of the form $A(x)$, or (ii) there is an r-successor y of x, such that $q_y(y)$ does not admit a homomorphism to $(\mathcal{A}_\varphi, b_{i,1})$ for any i with $1 \leq i \leq n$. Using the definition of $\mathcal{A}_\varphi$, we can see that (i) cannot happen because it implies that q does not fit the positive examples in E_φ. Therefore, case (ii) must apply. Let y be the r-successor in question.

We know that $\mathcal{A}_\varphi, \varnothing \not\models q_y(b_{i,1})$ for all i with $1 \leq i \leq n$. Furthermore, since q fits the positive examples in E_φ, for each i, either $\mathcal{A}_\varphi, \varnothing \models q_y(p_{i,1})$ or $\mathcal{A}_\varphi, \varnothing \models q_y(n_{i,1})$. Now, let L_y be the set

$$\{l \in L \mid A(z) \in q_y \text{ with } \mathsf{dist}_q(y, z) + 1 = j_l\}.$$

Since q fits the positive examples, L_y does not contain both X_i and $\neg X_i$ for any i. Suppose, for the sake of a contradiction, that φ has a clause φ_i, such that no literal occurring in φ_i belongs to L_y. Then, $\mathcal{A}_\varphi, \varnothing \models q_y(b_{i,1})$, as witnessed by the homomorphism that maps each variable z to $b_{i,k}$ where $k = \mathsf{dist}_q(y, z) + 1$. However, we know that $\mathcal{A}_\varphi, \varnothing \not\models q_y(b_{i,1})$, a contradiction. Hence, L_y contains a literal from every clause of φ, and every assignment that is consistent with L_y satisfies φ. □

From Lemma 6.5, together with the NP-hardness of 3CNF satisfiability, we immediately obtain the following.

Lemma 6.6. *Let $\mathbf{Q}$ be any class of unary CQs that contains all path CQs, and let $\mathcal{L}$ be any ontology language. The fitting problem for $\mathbf{Q}$ and $\mathcal{L}$ on $M = \{E_\varphi \mid \varphi \text{ is a 3CNF formula}\}$ is NP-hard.*

Now, we can apply Lemma 6.1.

Theorem 6.7. *Let $\mathbf{Q}$ be any class of unary CQs that contains all path CQs over a fixed signature with at least one role name and one concept name. Then,*

1. *$\mathbf{Q}$ is not polynomial time PAC learnable under any ontology language, and*
2. *$\mathbf{Q}$ queries are not polynomial time learnable using only CQ-equivalence queries,*

unless RP = NP.

Proof. Assume that $\mathbf{Q}$ is polynomial PAC learnable under the empty ontology language. Let $M = \{E_\varphi \mid \varphi \text{ is a 3CNF formula}\}$. Then, by Lemma 6.3, $\mathbf{Q}$ queries can be answered in polynomial time on the examples that occur in M, since all examples are tree-shaped. Additionally, it follows from Lemma 6.5 that $\mathbf{Q}$ has the polynomial fitting property on M. Therefore, the fitting problem for $\mathbf{Q}$ on M is in RP by Lemma 6.1. Since the fitting problem for $\mathbf{Q}$ on M is NP-hard by Lemma 6.5, it follows that RP = NP.

For Point 2, recall that an exact learning algorithm that uses only equivalence queries can be used to construct a PAC learning algorithm by Theorem 3.14. □

Note that ELQs, ELIQs and CQs contain all path CQs. Moreover, Point 2 of Theorem 6.7 can be strengthened to "unless P = NP" if shown directly and not via the connection to PAC learning. See [FJL21a, Theorem 2].

In Chapters 4 and 5 we have seen that ELQs, ELIQs and CQs are polynomial time learnable under the empty ontology if the learning algorithm can use both membership queries and equivalence queries (Theorem 4.42, Theorem 5.39, Proposition 5.3). Now, Theorem 6.7 implies that membership queries are indeed necessary for polynomial time learning and cannot be omitted. Furthermore, the results in Section 4.1 imply that it is also not possible to avoid equivalence queries.

Motivated by these results, we turn our focus to *sample-efficient* PAC learning algorithms, that do not need to run in polynomial time.

6.2 Sample-Efficient PAC Learning of Queries

As there exists no polynomial time PAC learning algorithm for ELQs by Theorem 6.7, we aim to determine if there are *sample-efficient* PAC learning algorithms for ELQs. Recall that, per Definition 3.13, a PAC learning algorithm is sample-efficient, if its sample size m is a polynomial of $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, the size of the signature $|\Sigma|$, the size of the ontology $\|\mathcal{O}\|$, the size of the target query $\|q_T\|$, and the maximal size of the examples.

In this section, we show that every fitting algorithm that produces *small hypotheses* is a sample-efficient PAC learning algorithm. Intuitively, this makes sense for the reason that fitting algorithms that can output hypotheses of unrestricted size can include information about every labeled example in their hypothesis. The hypotheses can therefore just memorize the labels in the sample, and do not necessarily generalize to unseen examples. In contrast, fitting algorithms that output hypotheses of small size need to compress the information available in the sample, and therefore need to find commonalities of the examples and generalize to unseen examples. Indeed, restricting the size of the output hypothesis to grow sublinearly with the sample size suffices for fitting algorithms to be sample-efficient PAC learning algorithms.

The connection between fitting algorithms that produce small hypotheses and sample-efficient PAC learning algorithms is well known, see for example [Blu+89] or [SB14]. We follow Blumer et al. [Blu+89] and formalize this connection for our setting in terms of Occam algorithms and the Vapnik-Chervonenkis dimension (VC-dimension).

Occam Algorithms and the VC-Dimension

In our learning setting, the VC-dimension measures the expressive power of a query class. Let $\mathcal{O}$ be an ontology and $\mathcal{Q}$ a query class. We say that $\mathcal{Q}$ *shatters* a set of data examples S under $\mathcal{O}$ if for every subset $S' \subseteq S$, there is a $q \in \mathcal{Q}$ such that $S' = \{(\mathcal{A}, \overline{a}) \in S \mid \mathcal{A}, \mathcal{O} \models q(\overline{a})\}$. The *VC-dimension* of $\mathcal{Q}$ under $\mathcal{O}$ is the cardinality of the largest set of examples S that is shattered by $\mathcal{Q}$ under $\mathcal{O}$. Note that if $\mathcal{Q}$ is a finite query class, then the largest set $\mathcal{Q}$ can shatter has cardinality $|S| = \log|\mathcal{Q}|$, as $\mathcal{Q}$ needs to contain a query for all $2^{|S|}$ subsets of S. Additionally, the presence of non-empty ontologies *does not increase* the VC-dimension of a query class, as concept inclusions only increase the number of queries that are equivalent.

Definition 6.8 (Occam algorithm)**.** Let $\mathcal{L}$ be an ontology language, $\mathcal{Q}$ a class of queries, and $\mathbf{A}$ a fitting algorithm for $\mathcal{Q}$ and $\mathcal{L}$. For an $\mathcal{L}$ ontology $\mathcal{O}$, a finite signature Σ, and $n_{q_T}, m \geq 1$, the *effective hypothesis space* $H^{\mathbf{A}}(\mathcal{O}, \Sigma, n_{q_T}, m)$ of $\mathbf{A}$ is the set of all outputs that $\mathbf{A}$ makes when started on $\mathcal{O}$ and a collection of m examples E over Σ that are labeled according to q_T under $\mathcal{O}$ for some $q_T \in \mathcal{Q}_\Sigma$ with $\|q_T\| \leq n_{q_T}$.

The fitting algorithm $\mathbf{A}$ is an *Occam algorithm* if there exists a polynomial p and a constant α with $0 \leq \alpha < 1$ such that for all $\mathcal{L}$ ontologies $\mathcal{O}$, finite signatures Σ, and $n_{q_T}, m \geq 1$, the VC-dimension of $H^{\mathbf{A}}(\mathcal{O}, \Sigma, n_{q_T}, m)$ under $\mathcal{O}$ is bounded from above by $p(n_{q_T}, |\Sigma|) \cdot m^\alpha$.

Theorem 3.2.1 of [Blu+89] implies the following, where log denotes the binary logarithm.

Lemma 6.9. *If* $\mathbf{A}$ *is an Occam algorithm with the VC-dimension of effective hypothesis spaces bounded by* $p(n_{q_T}, |\Sigma|) \cdot m^\alpha$*, then* $\mathbf{A}$ *is a PAC learning algorithm with sample size*

$$m\left(\frac{1}{\epsilon}, \frac{1}{\delta}, |\Sigma|, \|\mathcal{O}\|, n_{q_T}, n_{\mathcal{A}}\right) := \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \left(\frac{8p(n_{q_T}, |\Sigma|)}{\epsilon}\log\frac{13}{\epsilon}\right)^{1/(1-\alpha)}\right).$$

There are differences between the setting in this thesis and the setup of Blumer et al. We comment on why Lemma 6.9 follows from Theorem 3.2.1 of [Blu+89]: Blumer et al. study the learning of *concept classes*, which are defined as a set C of *concepts* $C \subseteq X$ where X is a fixed set of *examples*. Consequently, their definition of PAC algorithms refers to concept classes and, in contrast to Definition 3.13, does neither mention ontologies nor signatures. However, when fixing an $\mathcal{L}$ ontology $\mathcal{O}$ and signature Σ, we obtain an associated concept class $C_{\mathcal{O},\Sigma}$ by taking X to be the set of all data examples over Σ and each query $q \in \mathcal{Q}_\Sigma$ as the concept that consists of all data examples that are positive examples for q under $\mathcal{O}$. Moreover, by fixing $\mathcal{O}$ and Σ, any fitting algorithm $\mathbf{A}$ for $\mathcal{Q}$ and $\mathcal{L}$ turns into a learning algorithm for $C_{\mathcal{O},\Sigma}$ in the sense of Blumer et al. Here, *fixing* means that we promise to only run $\mathbf{A}$ on

input ontology $\mathcal{O}$ and collections of labeled examples E that are labeled according to some $q_T \in \mathcal{Q}_\Sigma$ under $\mathcal{O}$.

In contrast to Definition 6.8, the definition of Occam algorithms of Blumer et al. refers to effective hypothesis spaces $H^{\mathbf{A}}(s, m)$ and requires that their VC-dimension is bounded by $p(s) \cdot m^\alpha$, where $\|\mathcal{O}\|$ and $|\Sigma|$ are considered constants. If $\mathbf{A}$ is an Occam algorithm in the sense of Definition 6.8, then $\mathbf{A}$ with $\mathcal{O}$ and Σ fixed is an Occam algorithm as defined by Blumer et al. The Theorem 3.2.1 of Blumer et al. then implies that $\mathbf{A}$ with $\mathcal{O}$ and Σ fixed is a PAC learning algorithm for $C_{\mathcal{O},\Sigma}$ with the bound stated in Lemma 6.9. Then, every fitting algorithm $\mathbf{A}$ that is a PAC learning algorithm when restricted to $\mathcal{O}$ and Σ, for *any* specific $\mathcal{O}$ and Σ and with the *same function* m describing the sample size, is a PAC learning algorithm for $\mathcal{L}$ and $\mathcal{Q}$.

There are two more small differences between our setup and the one of Blumer et al. First, one of the preconditions of Theorem 3.2.1 is that an Occam algorithm runs in polynomial time, but an analysis of the proof shows that this assumption is not used. Second, the sample size m in the definition of PAC algorithms of Blumer et al. does not depend on the size of the examples. Definition 3.13 is a standard variation which does not impair the application of Theorem 3.2.1. To see this, it suffices to observe that we do not use this parameter in the definition of effective hypothesis spaces and thus Occam algorithms (with fixed $\mathcal{O}$ and Σ) according to Definition 6.8 are also Occam algorithms in the sense of Blumer et al. Moreover, every PAC algorithm in the sense of Blumer et al. is a PAC algorithm according to Definition 3.13.

Including the data example size as a parameter of the sample size m strengthens results which prove that algorithms are not sample-efficient PAC learning algorithms, as it makes it impossible to use data examples of excessive size. It is also more generous regarding the upper bounds (developing PAC algorithms), but we do not make use of that generosity.

A perhaps surprising observation is that following Definition 3.13, due to the generosity of including $n_{\mathcal{A}}$ as a parameter of the sample size, every fitting algorithm is a (not necessarily sample-efficient) PAC learning algorithm. This is because the probability distribution P from which examples are drawn may only assign a non-zero probability to examples with $\|\mathcal{A}\| \leq n_{\mathcal{A}}$, which means the number of possible examples is bounded exponentially by $n_{\mathcal{A}}$. Hence, the sample size m can be chosen, such that every sample contains, with high likelihood, a large portion of the possible examples, and every query that fits the sample therefore has only a small error over the entire distribution. This, of course, requires that m is an exponential function, and hence does not show sample-efficient PAC learnability.

Algorithm 6.1: The bounded fitting algorithm for $\mathcal{L}$ and $\mathcal{Q}$.

Parameter A size-restricted fitting algorithm $\mathbf{A}$ for $\mathcal{L}$ and $\mathcal{Q}$
Input A signature Σ, an $\mathcal{L}$ ontology $\mathcal{O}$, and labeled data examples E
Output A $\mathcal{Q}_\Sigma$ query that fits E under $\mathcal{O}$

$n_q := 1$
while $\mathbf{A}(\Sigma, \mathcal{O}, E, n_q)$ does not return a fitting query **do**
    $n_q := n_q + 1$
end while
return the result of $\mathbf{A}(\Sigma, \mathcal{O}, E, n_q)$

Bounded Fitting

Lemma 6.9 tells us that we can obtain a sample-efficient PAC learning algorithm for a query class from an Occam algorithm. To formulate an Occam algorithm, it is important to realize that the VC-dimension of an effective hypothesis space is bounded by the *cardinality* of the effective hypothesis space, and therefore also by the *size* of hypotheses that an algorithm outputs. This is because, for a fixed signature Σ and alphabet, the number of queries q with $\|q\| \leq n$ is only exponential in n.

Indeed, we show that an algorithm that outputs the smallest fitting query is an Occam algorithm for every query class $\mathcal{Q}$ and ontology language $\mathcal{L}$. For this, we define a variant of the fitting problem, namely size-restricted fitting. A *size-restricted fitting algorithm* for $\mathcal{L}$ and $\mathcal{Q}$ takes as input a signature Σ, an $\mathcal{L}$ ontology $\mathcal{O}$, a collection of labeled data examples E and a natural number $n_q \geq 1$ in unary and returns, if it exists, a $q \in \mathcal{Q}_\Sigma$ with $\|q\| \leq n_q$ that fits E under $\mathcal{O}$. If no such query exists, it returns *no*. Note that in contrast to fitting algorithms, we demand size-restricted fitting algorithms to always terminate, which is not a difficult requirement, as the set of queries $q \in \mathcal{Q}_\Sigma$ with $\|q\| \leq n_q$ is finite and can be enumerated.

The simple principle of a *bounded fitting algorithm* is displayed in Algorithm 6.1. A bounded fitting algorithm uses an algorithm for the size-restricted fitting problem, to identify the smallest fitting query for labeled data examples. Inspired by approaches such as *bounded model checking*, it first searches for a fitting of size 1, then of size 2, and so on, until one is found, and then returns it. Note that a bounded fitting algorithm does not terminate if there is no fitting query, but in practice size bounds can be used to guarantee termination. Of course, the smallest fitting query can also be found by the arguably simpler strategy of enumerating all queries ordered by size. However, size-restricted fitting algorithms may use smarter strategies than naive enumeration and thus be significantly faster.

Theorem 6.10. *Let $\mathcal{L}$ be an ontology language and $\mathbf{Q}$ a query class. Every bounded fitting algorithm for $\mathcal{L}$ and $\mathbf{Q}$ is a sample-efficient PAC learning algorithm with sample size in*

$$O\Big(\frac{1}{\epsilon} \cdot \log \frac{1}{\epsilon} \cdot \log \frac{1}{\delta} \cdot \log|\Sigma| \cdot \|q_T\|\Big).$$

Proof. Let $\mathbf{B}$ be a bounded fitting algorithm for $\mathcal{L}$ and $\mathbf{Q}$. Let $\mathcal{O}$ be an $\mathcal{L}$ ontology, Σ a signature, and $n_{q_T}, m \geq 0$. We show that the VC-dimension of $H^{\mathbf{B}}(\mathcal{O}, \Sigma, n_{q_T}, m)$ is at most $O(n_{q_T} \cdot \log|\Sigma|)$.

It is immediate from Algorithm 6.1 that, when started on Σ, $\mathcal{O}$ and a collection of m data examples E, that is labeled according to some $q_T \in \mathbf{Q}_\Sigma$ with $\|q_T\| \leq n_{q_T}$ under $\mathcal{O}$, then $\mathbf{B}$ returns a $q \in \mathbf{Q}_\Sigma$ that fits E under $\mathcal{O}$ whose size $\|q\|$ is smallest among all fitting queries. Consequently, $H^{\mathbf{B}}(\mathcal{O}, \Sigma, n_{q_T}, m)$ consists only of queries $q \in \mathbf{Q}_\Sigma$ with $\|q\| \leq n_{q_T}$. There are at most $(|\Sigma| + c + 1)^{n_{q_T}}$ such queries for some constant[1] c and since n queries can at most shatter a set of cardinality $\log n$, the VC-dimension of $H^{\mathbf{B}}(\mathcal{O}, \Sigma, n_{q_T}, m)$ is at most

$$\log\,(|\Sigma| + c + 1)^{n_{q_T}} \in O(n_{q_T} \cdot \log|\Sigma|),$$

as desired. It remains to apply Lemma 6.9. □

It is interesting that the sample size in Theorem 6.10 does not depend on the size of the examples $n_{\mathcal{A}}$ or the size of the ontology $\|\mathcal{O}\|$. This means that bounded fitting is even a sample-efficient PAC learning algorithm in a stricter sense than Definition 3.13. Additionally, it is useful to note that Theorem 6.10 does not rely on the exact size measure used in the size-restricted fitting algorithm. If we choose a different one, like the number of variables as a size measure for ELQs, we can obtain a similar result, by using the following in the proof of Theorem 6.10.

For $n \geq 1$, let $\mathrm{ELQ}^{\exists}_{(n)}$ be the set of ELQs that use at most n variables.

Lemma 6.11. *Let $\mathcal{L}$ be an ontology language and $\mathcal{O}$ an $\mathcal{L}$ ontology. Then, for all $n \geq 1$, the VC-dimension of $\mathrm{ELQ}^{\exists}_{(n)}$ under $\mathcal{O}$ is at most $2(|\Sigma| + 1)n$.*

Proof. We first observe that the number of queries in $\mathrm{ELQ}^{\exists}_{(n)}$ is bounded from above by $m_n = 4^{(|\Sigma|+1)n}$. To see this, note that the number of rooted, directed, unlabeled trees with n nodes is bounded from above by the n-th Catalan number [OEI24], which, in turn, is bounded from above by 4^n [DB86]. Each such tree gives rise to an ELQ by assigning a unique role name from Σ to each of the at most $n - 1$ edges

[1] The number of symbols from the finite alphabet used to encode syntactic objects as a word from the definition of $\|\cdot\|$.

of the tree and a set of concept names from Σ to each of the at most n nodes of the tree. This clearly yields the stated bound m_n as

$$4^n \cdot |\Sigma|^{n-1} \cdot |\Sigma|^n \leq 4^{(|\Sigma|+1)n}.$$

Then, the VC-dimension of $\mathrm{ELQ}^{\exists}_{(n)}$ under the empty ontology is at most $\log m_n = 2(|\Sigma| + 1)n$. Making the ontology non-empty may only decrease the VC-dimension as it may make non-equivalent concepts equivalent, but not vice versa. □

If one views other query classes, like ELIQs or $\mathcal{ALC}$ concepts, as syntax trees, one can see that the proof of Lemma 6.11 also applies to them. We use this different size bound in Section 6.4.

6.3 Not Sample-Efficient Fitting Algorithms

Although it is straightforward to obtain sample-efficient PAC learning algorithms for a query class by using bounded fitting, not every fitting algorithm is sample-efficient. Indeed, fitting algorithms that do not aim to find a fitting query of minimal size but aim for other properties of the fitting can often be shown not to be sample-efficient PAC learning algorithms.

Under certain assumptions, it is known that sample-efficient PAC learning implies the existence of Occam algorithms [BP92]. Meaning that, sample-efficient PAC learning is intrinsically linked to finding small fitting queries. These assumptions, however, require the concept space to be *polynomially closed under exception lists*, which is not known to be the case for queries under ontologies.

Recently, *extremal fitting CQs* have been studied by [tCat+23b]. These are CQs that fit given examples, but also have logically interesting properties in the set of all fitting queries. In this section, we show that fitting algorithms that produce these kinds of extremal fittings are *not sample-efficient PAC learning algorithms*. This already holds in the case without an ontology. To simplify writing, we therefore drop saying *under the empty ontology*. Although we show these results for CQs, the same also holds for ELQs and ELIQs [tCat+23c].

Additionally, we show that fitting algorithms that aim to produce ELQs of minimal quantifier depth, are also not sample-efficient PAC learning algorithms.

We begin by formally introducing extremal fittings. Let E be a collection of labeled examples. A CQ q that fits E is a

- *most-specific fitting CQ* if for every CQ q' that fits E, $q \subseteq_\varnothing q'$;
- *strongly most-general fitting CQ* if for every CQ q' that fits E, $q' \subseteq_\varnothing q$;

- *weakly most-general fitting CQ* if for every CQ q' that fits E, $q \subseteq_\varnothing q'$ implies $q \equiv_\varnothing q'$.

There is also a more general notion than strongly most-general fitting:

- a finite set of CQs $\{q_1, \dots, q_n\}$ is a *basis of most-general fitting CQs* for E if each q_i fits E and for all CQs q' that fit E, we have $q' \subseteq_\varnothing q_i$ for some $i \leq n$.

Note that the definition of most-specific fitting CQs parallels that of strongly most-general fitting CQs. One may also define a weak version of most-specific fitting CQs, but this definition turns out to be equivalent to the above definition of most-specific fitting CQ [tCat+23b]. The different notions of most-general fitting CQ are connected as follows. If q is a strongly most-general fitting CQ, then $\{q\}$ is a basis of most-general fitting CQs. If a basis of most-general fitting CQs is subset minimal, then it contains only weakly most-general fitting CQs. Furthermore, if a strongly most-general fitting CQ exists, then it is also the only weakly most-general fitting CQ.

A most-specific fitting CQ q and a basis of most-general fitting CQs $\{q_1, \dots, q_n\}$ completely describe the space of all fitting CQs: For all CQs q', $q \subseteq_\varnothing q'$ and $q' \subseteq_\varnothing q_i$ for some i, if and only if q' fits E.

Example 6.12. Consider the positive example $(\{A_1(a), A_2(a), A_3(a)\}, a)$ and the negative examples $(\{A_1(b)\}, b), (\{A_2(b)\}, b)$. The ELQ $q(x) \leftarrow A_1(x) \wedge A_2(x) \wedge A_3(x)$ is the most-specific fitting CQ of these examples. The ELQs $q_1(x) \leftarrow A_1(x) \wedge A_2(x)$ and $q_2(x) \leftarrow A_3(x)$ are both weakly most-general fitting CQs. There is no strongly most-general fitting CQ, but the set $\{q_1, q_2\}$ is a basis of most-general fitting CQ.

Most-specific fitting CQs can be characterized in terms of direct products.

Theorem 6.13 ([tCat+23b])**.** *For all CQs q and collections of labeled examples E, the following are equivalent:*

1. *q is a most-specific fitting CQ for E,*
2. *q fits E and is equivalent to the canonical CQ of $\prod_{(\mathcal{A},\bar{a},+)\in E} \mathcal{A}$.*

Strongly most-general fitting CQs and finite bases of most-general fitting CQs can be characterized in terms of *homomorphism dualities*, a fundamental concept that originates from combinatorial graph theory and that has found diverse applications in different areas, including the study of constraint satisfaction problems, database theory and knowledge representation. Here, we use a relativized version of homomorphism dualities introduced by ten Cate et al. [tCat+23b].

Definition 6.14 (Relativized homomorphism duality)**.** Let $(\mathcal{B}, \bar{b})$ be an example. A *homomorphism duality relative to* $(\mathcal{B}, \bar{b})$ is a pair of finite sets of examples[2] (F, D) such that for all examples $(\mathcal{B}', \bar{b}')$ with $\mathcal{B}', \bar{b}' \to \mathcal{B}, \bar{b}$, the following are equivalent:

1. $\mathcal{A}_f, \bar{a}_f \to \mathcal{B}', \bar{b}'$ for some $(\mathcal{A}_f, \bar{a}_f) \in F$,
2. $\mathcal{B}', \bar{b}' \not\to \mathcal{A}_d, \bar{a}_d$ for all $(\mathcal{A}_d, \bar{a}_d) \in D$.

The use of *relativized* in Definition 6.14 refers to the restriction to only examples $(\mathcal{B}', \bar{b}')$ with $\mathcal{B}', \bar{b}' \to \mathcal{B}, \bar{b}$. If one instead considers all examples $(\mathcal{B}', \bar{b}')$ one obtains the more common notion of *unrelativized* homomorphism dualities, which are strongly connected (in the case without an ontology) to the unique characterizations and frontiers we encountered before [tCD22]. Most-general fittings can be characterized with the relativized variant as follows.

Theorem 6.15 ([tCat+23b])**.** *For all CQs $q_1, \dots, q_n$ and collections of labeled examples E, the following are equivalent:*

1. *$\{q_1, \dots, q_n\}$ is a basis of most-general fitting CQs for E*
2. *each q_i fits E, and $(\{(\mathcal{A}_{q_1}, \bar{a}_{q_1}), \dots, (\mathcal{A}_{q_n}, \bar{a}_{q_n})\}, \{(\mathcal{B}, \bar{b}) \mid (\mathcal{B}, \bar{b}, -) \in E\})$ is a homomorphism duality relative to $\prod_{(\mathcal{A}, \bar{a}, +) \in E} \mathcal{A}$.*

Or, formulated for strongly most-general fitting CQs: q is a strongly most-general fitting CQ if and only if q fits E and $(\{(\mathcal{A}_q, \bar{a}_q)\}, \{(\mathcal{B}, \bar{b}) \mid (\mathcal{B}, \bar{b}, -) \in E\})$ is a homomorphism duality relative to $\prod_{(\mathcal{A}, \bar{a}, +)} \mathcal{A}$.

Most-General Fittings Preclude Sample-Efficiency

We show that fitting algorithms that always produce a most-general fitting CQ, if it exists, cannot be sample-efficient PAC learning algorithms. We only show this for strongly most-general fitting CQs since fitting algorithms that always produce a weakly most-general fitting CQ or the elements of a base of most-general fitting CQs must also produce a strongly most-general fitting CQ if it exists. Consequently, our result also applies to the latter kinds of algorithms. Guided by Theorem 6.15 we base our proofs on relativized homomorphism dualities.

Known constructions of a homomorphism duality (F, D) from F relative to some example $(\mathcal{B}, \bar{b})$ result in examples D with size exponential in $\|F\|$ and $\|\mathcal{B}\|$, already for acyclic examples [tCD22; NT05]. This makes them unsuitable for showing

[2] The F stands for *forbidden* (as dualities were introduced in the context of forbidding certain graph patterns) and the D for *duals*.

non-sample-efficiency, as it would force us to use negative data examples that are exponentially larger than the positive examples. Since Definition 3.13 allows the sample size m to depend on the size of the examples, examples of exponential size would effectively allow a sample-efficient PAC learning algorithm to use an exponential number of examples. In order to avoid this effect, we begin by showing that for a certain restricted class of examples F, a relativized homomorphism duality (F, D) exists such that D is of polynomial size and can be computed in polynomial time.

A *path ABox* is an ABox $\mathcal{A}$ such that the assertions in $\mathcal{A}$ are of the form

$$r_1(a_0, a_1), \dots, r_n(a_{n-1}, a_n), A_1(a_{j_1}), \dots, A_m(a_{j_m})$$

where all r_i are role names, all A_i are concept names, and for all j_i, $j_i > 0$. A *path example* $(\mathcal{A}, a)$ is a unary data example, where $\mathcal{A}$ is a path ABox and $a = a_0$. This is the equivalent of *path CQs* from Section 6.1 for data examples, except that multiple different concept names and role names are permitted.

Lemma 6.16. *Let $(\mathcal{A}, a_0)$ and $(\mathcal{B}, b_0)$ be path examples. There exists an example $(\widehat{\mathcal{B}}, \widehat{b})$ that can be computed in time polynomial in $\|\mathcal{A}\|$ and $\|\mathcal{B}\|$ such that $(\{(\mathcal{A}, a_0)\}, \{(\widehat{\mathcal{B}}, \widehat{b})\})$ is a homomorphism duality relative to $(\mathcal{B}, b_0)$.*

Proof. Since $\mathcal{A}$ and $\mathcal{B}$ are path ABoxes, assume that $\mathsf{ind}(\mathcal{A}) = \{a_0, \dots, a_n\}$ and $\mathsf{ind}(\mathcal{B}) = \{b_0, \dots b_m\}$. We can check in polynomial time whether $\mathcal{A}, a_0 \to \mathcal{B}, b_0$. We distinguish cases.

If $\mathcal{A}, a_0 \not\to \mathcal{B}, b_0$, then it follows that $\mathcal{A}, a_0 \not\to \mathcal{B}', b'$ for all data examples $\mathcal{B}', b'$ with $\mathcal{B}', b' \to \mathcal{B}, b_0$. Then, $(\{(\mathcal{A}, a_0)\}, \{(\mathcal{B}, b_0)\})$ is a homomorphism duality relative to $(\mathcal{B}, b_0)$.

If $\mathcal{A}, a_0 \to \mathcal{B}, b_0$, we construct a data example $(\widehat{\mathcal{B}}, \widehat{b})$ such that $(\{(\mathcal{A}, a_0)\}, \{(\widehat{\mathcal{B}}, \widehat{b})\})$ is the desired homomorphism duality relative to $(\mathcal{B}, b_0)$ as follows. Since $\mathcal{A}, a_0 \to \mathcal{B}, b_0$, it must be that $n \leq m$ and the role name r_i is the same in $\mathcal{A}$ and $\mathcal{B}$ for all i with $1 \leq i \leq n$. The individual names used in $\widehat{\mathcal{B}}$ are pairs $\langle b_i, f \rangle$ where $b_i \in \mathsf{ind}(\mathcal{B})$ and f is an assertion from $\mathcal{A}$ that mentions a_i or the dummy assertion $\circ$. More specifically, $\mathsf{ind}(\widehat{\mathcal{B}})$ is the following set of individual names:

$$\begin{aligned}
&\{\langle b_i, \circ \rangle \mid i = 0 \text{ or } n < i \leq m\} \cup \\
&\{\langle b_i, r_i(a_{i-1}, a_i) \rangle \mid 1 \leq i \leq n\} \cup \\
&\{\langle b_i, r_{i+1}(a_i, a_{i+1}) \rangle \mid 0 \leq i \leq n-1\} \cup \\
&\{\langle b_i, A(a_i) \rangle \mid 1 \leq i \leq n \text{ and } A(a_i) \in \mathcal{A}\}.
\end{aligned}$$

For all individual names $\langle b_i, f \rangle, \langle b_{i+1}, f' \rangle$ in the set above, we add to $\widehat{\mathcal{B}}$ the assertion

- $r_{i+1}(\langle b_i, f \rangle, \langle b_{i+1}, f' \rangle)$ if $f \neq r_{i+1}(a_i, a_{i+1})$ or $f' \neq r_{i+1}(a_i, a_{i+1})$,

- $A(\langle b_{i+1}, f' \rangle)$ if $A(b_{i+1}) \in \mathcal{B}$ and $f' \neq A(a_{i+1})$.

Then, set $\widehat{b}$ to be the individual name $\langle b_0, R_1(a_0, a_1) \rangle$. This completes the construction of $(\widehat{\mathcal{B}}, \widehat{b})$. Note that every role assertion $r_{i+1}(\langle b_i, f \rangle, \langle b_{i+1}, f' \rangle)$ in $\widehat{\mathcal{B}}$ is a copy of a role assertion $r_{i+1}(b_i, b_{i+1})$ in $\mathcal{B}$. Since $f, f' \in \mathcal{A} \cup \{\circ\}$, there are at most $(\|\mathcal{A}\| + 1)^2$ copies of every role assertion of $\mathcal{B}$ in $\widehat{\mathcal{B}}$. The same is true for concept name assertions. Therefore, $\|\widehat{\mathcal{B}}\| \leq \|\mathcal{B}\| \cdot (\|\mathcal{A}\| + 1)^2$. It is easy to see that $\widehat{\mathcal{B}}$ can be computed in polynomial time using the above construction.

It remains to show that $(\{(\mathcal{A}, a_0)\}, \{(\widehat{\mathcal{B}}, \widehat{b})\})$ is a homomorphism duality relative to $(\mathcal{B}, b_0)$. First, we prove that $\mathcal{A}, a_0 \to \mathcal{B}', b'$ implies $\mathcal{B}', b' \nrightarrow \widehat{\mathcal{B}}, \widehat{b}$ for all examples $(\mathcal{B}', b')$ with $\mathcal{B}', b' \to \mathcal{B}, b_0$. For this, it suffices that $\mathcal{A}, a_0 \nrightarrow \widehat{\mathcal{B}}, \widehat{b}$. For all k, let $\mathcal{A}_k$ be the restriction of $\mathcal{A}$ to assertions that contain only values a_i with $i \geq k$. Then, for all i with $1 \leq i \leq n$ and all $\langle b_i, f \rangle \in \mathsf{ind}(\widehat{\mathcal{B}})$,

$$\mathcal{A}_i, a_i \to \widehat{\mathcal{B}}, \langle b_i, f \rangle \quad \text{implies} \quad f = r_i(a_{i-1}, a_i). \tag{$*$}$$

We show this by induction on $n-i$. In the induction start, let $i = n$. By construction of $\widehat{\mathcal{B}}$, $f \in \{r_n(a_{n-1}, a_n)\} \cup \{A(a_n) \mid A(a_n) \in \mathcal{A}_n\}$. Assume that $\mathcal{A}_n, a_n \to \widehat{\mathcal{B}}, \langle b_n, f \rangle$. Then, it follows that $A(\langle b_n, f \rangle) \in \widehat{\mathcal{B}}$ for all $A(a_n) \in \mathcal{A}$, hence $f \neq A(a_n)$ for any $A(a_n) \in \mathcal{A}$. Therefore, the only possibility for f that remains is that $f = r_n(a_{n-1}, a_n)$.

In the induction step, assume that ($*$) holds for $i+1$ and that $\mathcal{A}_i, a_i \to \widehat{\mathcal{B}}, \langle b_i, f \rangle$. Again, this implies that $A(\langle b_i, f \rangle) \in \widehat{\mathcal{B}}$ for all $A(a_i) \in \mathcal{A}_i$, hence $f \notin \{A(a_i) \mid A(a_i) \in \mathcal{A}_i\}$. Additionally, from $\mathcal{A}_i, a_i \to \widehat{\mathcal{B}}, \langle b_i, f \rangle$ and $i < n$ it follows that there must be a role assertion $r_{i+1}(\langle b_i, f \rangle, \langle b_{i+1}, f' \rangle) \in \widehat{\mathcal{B}}$ such that $\mathcal{A}_{i+1}, a_{i+1} \to \widehat{\mathcal{B}}, \langle b_{i+1}, f' \rangle$. By the induction hypothesis, $f' = r_{i+1}(a_i, a_{i+1})$. It follows from $r_{i+1}(\langle b_i, f \rangle, \langle b_{i+1}, f' \rangle) \in \widehat{\mathcal{B}}$, that $f \neq r_{i+1}(a_i, a_{i+1})$. Thus, the only possibility that remains is that $f = r_i(a_{i-1}, a_i)$.

From ($*$) and $r_1(a_0, a_1) \in \mathcal{A}$ it now follows that $\mathcal{A}, a_0 \nrightarrow \widehat{\mathcal{B}}, \langle b_0, 111(a_0, a_1) \rangle = \widehat{\mathcal{B}}, \widehat{b}$, since by construction of $\widehat{\mathcal{B}}$, $r_1(\langle b_0, r_1(a_0, a_1) \rangle, \langle b_1, r_1(a_0, a_1) \rangle) \notin \widehat{\mathcal{B}}$.

Second, we show that $\mathcal{A}, a_0 \nrightarrow \mathcal{B}', b'$ implies $\mathcal{B}', b' \to \widehat{\mathcal{B}}, \widehat{b}$ for all examples $(\mathcal{B}', b')$ with $\mathcal{B}', b' \to \mathcal{B}, b_0$. Let $\mathcal{B}', b'$ be an example such that $\mathcal{A}, a_0 \nrightarrow \mathcal{B}', b'$ and $\mathcal{B}', b' \to \mathcal{B}, b_0$. Let h be a homomorphism from $\mathcal{B}'$ to $\mathcal{B}$ with $h(b') = b_0$. We construct a homomorphism g from $\mathcal{B}'$ to $\widehat{\mathcal{B}}$ with $g(b') = \widehat{b}$ as follows. For all $a \in \mathsf{ind}(\mathcal{B}')$, there is an i such that $0 \leq i \leq m$ and $h(a) = b_i$. Define $g(a)$ depending on this i as follows. For $i = 0$,

- if $\mathcal{A}_i, a_i \to \mathcal{B}', a$, set $g(a) = \langle b_0, \circ \rangle$;
- otherwise, that is, if $\mathcal{A}_i, a_i \nrightarrow \mathcal{B}', a$, set $g(a) = \langle b_0, r_1(a_0, a_1) \rangle$.

For $1 \leq i \leq n$,

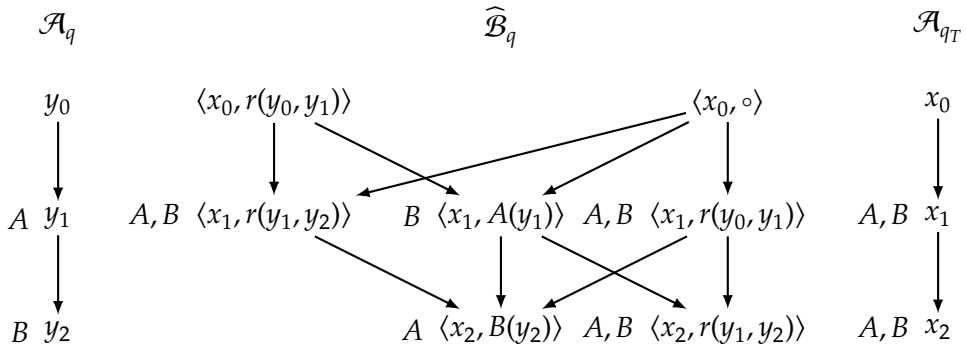

Figure 6.2: A homomorphism duality relative to a path example

- if $\mathcal{A}_i, a_i \to \mathcal{B}', a$, set $g(a) = \langle b_i, r_i(a_{i-1}, a_i)\rangle$;
- otherwise, that is, if $\mathcal{A}_i, a_i \nrightarrow \mathcal{B}', a$, we distinguish cases:
  - if there is an $A(a_i) \in \mathcal{A}_i$ such that $A(a) \notin \mathcal{B}'$, set $g(a) = \langle b_i, A(a_i)\rangle$;
  - otherwise, set $g(a) = \langle b_i, r_{i+1}(a_i, a_{i+1})\rangle$ (note that in this case $i < n$ and $\mathcal{A}_{i+1}, a_{i+1} \nrightarrow \mathcal{B}', a'$ for all $r_{i+1}(a, a') \in \mathcal{B}'$).

For $i > n$, set $g(a) = \langle b_i, \circ\rangle$.

It remains to verify that g is a homomorphism. Let $A(a)$ be a concept name assertion in $\mathcal{B}'$. Since h is a homomorphism, there is a fact $A(b_i) \in \mathcal{B}$ with $h(a) = b_i$. By definition of g, $g(a) = \langle b_i, f\rangle$ for some fact $f \in \{\circ, r_i(a_{i-1}, a_i), r_{i+1}(a_i, a_{i+1})\} \cup \{A'(a_i) \in \mathcal{A} \mid A' \neq A\}$. From the construction of $\widehat{\mathcal{B}}$ it follows that $A(g(a)) \in \widehat{\mathcal{B}}$ for all these cases of f.

Let $r(a, a')$ be a role assertion in $\mathcal{B}'$. Since h is a homomorphism, there is a role assertion $r(b_i, b_{i+1}) \in \mathcal{B}$ with $h(a) = b_i$, $h(a') = b_{i+1}$ and $r = r_{i+1}$. The function g then maps a to $\langle b_i, f\rangle$ and a' to $\langle b_{i+1}, f'\rangle$, for some assertions $f, f' \in \mathcal{A} \cup \{\circ\}$. It follows from the construction of $\widehat{\mathcal{B}}$ that there is an assertion $r(\langle b_i, f\rangle, \langle b_{i+1}, f'\rangle) \in \widehat{\mathcal{B}}$ for all f, f', except for $f = f' = r(a_i, a_{i+1}) \in \mathcal{A}$ (and $i + 1 \leq n$). Assume for contradiction that there is an assertion $r(a_i, a_{i+1}) \in \mathcal{A}$ and $f = f' = r(a_i, a_{i+1})$. By definition of g and $f = r(a_i, a_{i+1})$ it follows that there is no assertion $r(a, a'') \in \mathcal{B}'$ with $\mathcal{A}_{i+1}, a_{i+1} \to \mathcal{B}', a''$, and it follows from definition of g and $f' = r(a_i, a_{i+1})$ that $\mathcal{A}_{i+1}, a_{i+1} \to \mathcal{B}', a'$. A contradiction, since $r(a, a') \in \mathcal{B}'$. □

Example 6.17. Consider the path examples $(\mathcal{A}_q, y_0)$ and $(\mathcal{A}_{q_T}, x_0)$, and the ABox $\widehat{\mathcal{B}}_q$ displayed in Figure 6.2. The example $(\widehat{\mathcal{B}}_q, \langle x_0, r(y_0, y_1)\rangle)$ is the result of constructing a homomorphism duality of $(\mathcal{A}_q, y_0)$ relative to $(\mathcal{A}_{q_T}, x_0)$ according to the construction

in the proof of Lemma 6.16. Note that $\widehat{\mathcal{B}}_q$ is not itself a path ABox, and not even tree-shaped.

We now build on Lemma 6.16 and Theorem 6.15 to show that fitting algorithms that produce strongly most-general fittings are not sample-efficient PAC learning algorithms.

Theorem 6.18. *Let* **A** *be a fitting algorithm for CQs that always produces a strongly most-general fitting if it exists. Then* **A** *is not a sample-efficient PAC learning algorithm.*

Proof. Assume to the contrary that **A** is a sample-efficient PAC learning algorithm that produces a strongly most-general fitting CQ, if it exists, with associated polynomial sample size $m\colon \mathbb{R}^2 \times \mathbb{N}^4 \to \mathbb{N}$ as in Definition 3.13. We assume that the value of m is at least 2, for any input.

Choose a signature Σ that contains the concept names A, B and a role name r, $\delta = 0.5$, $\epsilon = 0.25$, and $n \in \mathbb{N}$ large enough so that

$$2^{n-1} > m\Big(\frac{1}{\delta}, \frac{1}{\epsilon}, |\Sigma|, 0, p_1(n), p_2(n)\Big).$$

In that bound, p_1 is a polynomial that bounds the size of the target CQ, and p_2 is a polynomial that bounds the size of the data examples, which we describe next. Assume without loss of generality that $p_2(n) \geq p_1(n)$. As target CQ, we use

$$q_T(x_0) \leftarrow r(x_0, x_1) \land A(x_1) \land B(x_1) \land \cdots \land r(x_{n-1}, x_n) \land A(x_n) \land B(x_n).$$

Thus, q_T is an r-path of length n in which every node after the first is labeled with A and B. We will use $(\mathcal{A}_{q_T}, x_0)$ as a positive example for q_T. Note that $\mathcal{A}_{q_T}$ is a path instance with $\|\mathcal{A}_{q_T}\| = p_1(n) \leq p_2(n)$.

Next, we construct instances that we use as negative examples. Define a set of CQs

$$S = \{q(y_0) \leftarrow r(y_0, y_1) \land \alpha_1(y_1) \land \cdots \land r(y_{n-1}, y_n) \land \alpha_n(y_n) \mid \alpha_i \in \{A, B\}\}.$$

The CQs in S resemble q_T, except that every node is labeled with only one of the concept names A and B. For all elements q of S it holds that $q_T \subseteq_\varnothing q$ and $\mathcal{A}_q$ is a path instance of polynomial size. In order to obtain the negative examples, we construct for each $q \in S$, the example $(\widehat{\mathcal{B}}_q, a_q)$ such that $(\{(\mathcal{A}_q, y_0)\}, \{(\widehat{\mathcal{B}}_q, a_q)\})$ is a homomorphism duality relative to $(\mathcal{A}_{q_T}, x_0)$. Using Lemma 6.16, it is easy to see that there is a fixed polynomial p_2 such that $\|\widehat{\mathcal{B}}_q\| \leq p_2(n)$. Since $\mathcal{A}_q, y_0 \to \mathcal{A}_{q_T}, x_0$ for all $q \in S$, it follows from the definition of relativized homomorphism duality (Definition 6.14) that $\mathcal{A}_{q_T}, x_0 \nrightarrow \widehat{\mathcal{B}}_q, a_q$. Hence, all $(\widehat{\mathcal{B}}_q, a_q)$ are negative examples for q_T. An example of this duality for $n = 2$ is displayed in Figure 6.2.

The central properties of the chosen examples are the following. For $q_1, q_2 \in S$, let $q_1 \wedge q_2$ denote the unary CQ that joins q_1 and q_2 at the answer variable y_0, but keeps all existential variables distinct. For $S' \subseteq S$, let $q_{S'} = \bigwedge_{q \in S} q$.

Claim. For every $S' \subseteq S$, $(\{(\mathcal{A}_{q_{S'}}, y_0)\}, \{(\widehat{\mathcal{B}}_q, a_q) \mid q \in S'\})$ is a homomorphism duality relative to $(\mathcal{A}_{q_T}, x_0)$.

Proof of the claim. Let S' be a subset of S. By Definition 6.14, we have to show that for all examples $(\mathcal{A}, a)$ with $\mathcal{A}, a \to \mathcal{A}_{q_T}, x_0$, it holds that $\mathcal{A}_{q_{S'}}, y_0 \to \mathcal{A}, a$ if and only if $\mathcal{A}, a \nrightarrow \widehat{\mathcal{B}}_q, a_q$ for all $q \in S'$.

Let $(\mathcal{A}, a)$ be an example such that $\mathcal{A}, a \to \mathcal{A}_{q_T}, x_0$. First, we show that $\mathcal{A}_{q'_S}, y_0 \to \mathcal{A}, a$ implies that $\mathcal{A}, a \nrightarrow \widehat{\mathcal{B}}_q, a_q$ for all $q \in S'$. So assume that $\mathcal{A}_{q'_S}, y_0 \to \mathcal{A}, a$. It suffices to show that $\mathcal{A}_{q_{S'}}, y_0 \nrightarrow \widehat{\mathcal{B}}_q, a_q$ for all $q \in S'$. Take any $q \in S'$. Since $(\{(\mathcal{A}_q, y_0)\}, \{(\widehat{\mathcal{B}}_q, a_q)\})$ is a homomorphism duality relative to $(\mathcal{A}_{q_T}, x_0)$ and $\mathcal{A}_q, y_0 \to \mathcal{A}_{q_T}, x_0$, we have that $\mathcal{A}_q, y_0 \nrightarrow \widehat{\mathcal{B}}_q, a_q$. By construction of $q_{S'}$, this implies $\mathcal{A}_{q_{S'}}, y_0 \nrightarrow \widehat{\mathcal{B}}_q, a_q$.

Next, we show that $\mathcal{A}_{q'_S}, y_0 \nrightarrow \mathcal{A}, a$ implies that there is a $q \in S'$ such that $\mathcal{A}, a \to \widehat{\mathcal{B}}_q, a_q$. Since $\mathcal{A}_{q_{S'}}$ is the join of all $\mathcal{A}_q$ with $q \in S'$ at the answer variable, there must be a $q \in S'$ such that $\mathcal{A}_q, y_0 \nrightarrow \mathcal{A}, a$. By definition of homomorphism dualities, therefore $\mathcal{A}, a \to \widehat{\mathcal{B}}_q, a_q$, as required. This completes the proof of the claim.

Let the probability distribution P assign probability 0.5 to the positive example $(\mathcal{A}_{q_T}, x_0)$, probability $\frac{1}{2^{n+1}}$ to all negative examples $(\widehat{\mathcal{B}}_q, a_q)$ with $q \in S$, and probability 0 to all other data examples. Now assume that the algorithm is started on a collection E of $m(\frac{1}{\delta}, \frac{1}{\epsilon}, |\Sigma|, 0, p_1(n), p_2(n))$ data examples drawn according to P and labeled according to q_T. Let $S' = \{q \in S \mid (\widehat{\mathcal{B}}_q, a_q, -) \in E\} \subseteq S$. We argue that $q_{S'}$ fits E. Since $\mathcal{A}_q, y_0 \to \mathcal{A}_{q_T}, x_0$ for all $q \in S$, it follows that $q_{S'}$ fits the positive examples in E which are all of the form $(\mathcal{A}_{q_T}, x_0)$. Now let $(\widehat{\mathcal{B}}_q, a_q)$ be a negative example in E. We have $\mathcal{A}_q, y_0 \nrightarrow \widehat{\mathcal{B}}_q, a_q$ by the properties of homomorphism dualities and by construction of $q_{S'}$ it follows that $q_{S'}(y_0) \nrightarrow \widehat{\mathcal{B}}_q, a_q$, as required.

We next observe that if $(\mathcal{A}_{q_T}, x_0, +) \in E$, then it follows from Theorem 6.15 and the claim that $q_{S'}$ is a strongly most-general fitting of E. Hence, with probability $1 - \frac{1}{2^{|E|}} > 1 - \delta$ (since the value of m and thus $|E|$ is at least 2), the algorithm **A** returns a CQ equivalent to $q_{S'}$.

We argue that $q_{S'}$ classifies all negative examples $(\widehat{\mathcal{B}}_q, a_q)$ with $q \in S \setminus S'$ incorrectly. To see this, let q be a CQ from $S \setminus S'$. Note that $\mathcal{A}_q, y_0 \nrightarrow \mathcal{A}_{q_{S'}}, y_0$ by construction of $q_{S'}$. Then, it follows from $(\{(\mathcal{A}_q, y_0)\}, \{(\widehat{\mathcal{B}}_q, a_q)\})$ being a homomorphism duality

relative to $(\mathcal{A}_{q_T}, x_0)$ and $\mathcal{A}_{q_{S'}}, y_0 \to \mathcal{A}_{q_T}, x_0$ that $\mathcal{A}_{q_{S'}}, y_0 \to \widehat{\mathcal{B}}_q, a_q$. Therefore, all $(\widehat{\mathcal{B}}_q, a_q)$ with $q \in S \setminus S'$ are labeled positively by $q_{S'}$.

Since $|S \setminus S'| = 2^n - |S'| > 2^n - 2^{n-1}$ by choice of n and each of the negative examples $(\widehat{\mathcal{B}}_q, a_q)$ with $q \in S \setminus S'$ has probability $\frac{1}{2^{n+1}}$ to be drawn from P, we have

$$\mathsf{error}_{P,q_T,\varnothing}(q_{S'}) > \frac{2^n - 2^{n-1}}{2^{n+1}} = \frac{1}{4} = \epsilon,$$

a contradiction. □

Most-Specific Fittings Preclude Sample-Efficient PAC Learning

In contrast to most-general fitting CQs, Theorem 6.13 implies that most-specific fitting CQs always exist, provided that a fitting CQ exists at all. We show that fitting algorithms that always produce a most-specific fitting CQ also cannot be sample-efficient PAC learning algorithms.

Theorem 6.19. *Let* **A** *be a fitting algorithm for CQs that always produces a most-specific fitting CQ. Then* **A** *is not a sample-efficient PAC learning algorithm.*

Proof. Assume to the contrary that **A** is a sample-efficient PAC learning algorithm with associated polynomial sample size $m\colon \mathbb{R}^2 \times \mathbb{N}^4 \to \mathbb{N}$ as in Definition 3.13. Choose a signature Σ that contains a concept name A and a role name r, set $q_T(x_0) \leftarrow A(x_0)$, $\delta = \epsilon = 0.5$, and n even and large enough such that

$$\frac{1}{2}\binom{n}{n/2} > m\left(\frac{1}{\epsilon}, \frac{1}{\delta}, |\Sigma|, 0, \|q_T\|, p(n)\right),$$

where p is a fixed polynomial that bounds the size of the examples that we are going to use. We next construct positive examples for q_T; negative examples are not needed. Let N denote the set of subsets of $\{1, \dots, n\}$ and let $N^{\frac{1}{2}}$ be defined likewise, but include only sets of cardinality exactly $n/2$. Note that $|N^{\frac{1}{2}}| = \binom{n}{n/2}$. With every $S \in N$, we associate the path ABox

$$\mathcal{A}_S = \{r(b_0, b_1), \dots, r(b_{n-1}, b_n)\} \cup \{A(b_i) \mid i \in S\}.$$

as well as the example $(\mathcal{A}'_S, a_0)$ that is obtained by constructing an example $(\mathcal{B}_S, a_0)$ such that the pair $(\{(\mathcal{A}_S, b_0)\}, \{(\mathcal{B}_S, a_0)\})$ is a homomorphism duality relative to $(\mathcal{A}_{\{1,\dots,n\}}, b_0)$ via the construction in the proof of Lemma 6.16 and then adding the fact $A(a_0)$. Due to this additional fact, every $(\mathcal{A}'_S, a_0)$ is a positive data example for q_T and using Lemma 6.16 one can show that there is a fixed polynomial p such that $\|\mathcal{A}'_S\| \leq p(n)$

Let P be the probability distribution that assigns probability $1/|N^{\frac{1}{2}}|$ to every example $(\mathcal{A}'_S, a_0)$ with $S \in N^{\frac{1}{2}}$, and probability 0 to all other examples. Now, assume that $\mathbf{A}$ is started on a collection of $m(\frac{1}{\epsilon}, \frac{1}{\delta}, |\Sigma|, 0, \|q_T\|, p(2n, 2n) + 1)$ data examples E drawn from P and labeled according to q_T, and let q_H be the CQ that is output by $\mathbf{A}$. Since E contains no negative examples, Theorem 6.13 implies that a most-specific fitting CQ exists. By the properties of $\mathbf{A}$, q_H must therefore be a most-specific CQ that fits E.

We argue that q_H labels incorrectly all data examples $(\mathcal{A}'_S, a_0)$ with $S \in N^{\frac{1}{2}}$ that are not in the sample E. Let S be any element of $N^{\frac{1}{2}}$ with $(\mathcal{A}'_S, a_0, +) \notin E$. We aim to show that $\mathcal{A}_{q_H}, a_{q_H} \nrightarrow \mathcal{A}'_S, a_0$. Let q_S be the canonical CQ of $(\mathcal{A}_S, b_0)$. Since $(\{(\mathcal{A}_S, b_0)\}, \{(\mathcal{B}_S, a_0)\})$ is a homomorphism duality relative to $(\mathcal{A}_{1,\dots,n}, b_0)$, and $\mathcal{A}'_S = \mathcal{B}_S \cup \{A(a_0)\}$, $\mathcal{A}_S, b_0 \nrightarrow \mathcal{A}'_S, a_0$. Hence, it suffices to show that $\mathcal{A}_S, b_0 \to \mathcal{A}_{q_H}, a_{q_H}$ or equivalently that $q_H \subseteq_\varnothing q_S$.

Recall that q_H is a most-specific fitting for E. Hence, it suffices to show that q_S is a fitting for E. Let S' be any element of $N^{\frac{1}{2}}$ with $(\mathcal{A}'_{S'}, a_0) \in E$. Then, $S \neq S'$ and thus $\mathcal{A}_{S'}, b_0 \nrightarrow \mathcal{A}_S, b_0$ by construction. Since $\mathcal{A}_S, b_0 \to \mathcal{A}_{\{1,\dots,n\}}, b_0$ and the pair $(\{(\mathcal{A}_{S'}, b_0)\}, \{(\mathcal{B}_{S'}, a_0)\})$ is a homomorphism duality relative to $(\mathcal{A}_{\{1,\dots n\}}, b_0)$, it follows that $\mathcal{A}_S, b_0 \to \mathcal{B}_{S'}, a_0$ and hence $\mathcal{A}_S, b_0 \to \mathcal{A}'_{S'}, a_0$. Therefore, $(\mathcal{A}'_{S'}, a_0)$ is a positive example for q_S, as required.

The definition of P and the choice of n now yield that with probability $1 > 1 - \delta$,

$$\mathsf{error}_{P,q_T}(q_H) = \frac{|N^{\frac{1}{2}}| - |E|}{|N^{\frac{1}{2}}|} > 0.5,$$

a contradiction. □

Minimum Quantifier Depth Precludes Sample-Efficiency

Other than most-specific and most-general fittings, one might also be interested in fitting queries that are *simple* in some sense, to aid understanding of their meaning. One way to judge the simplicity of queries, is the quantifier depth.

The *quantifier depth* of an ELQ is usually defined through its representation as an $\mathcal{EL}$ concept. There, it is simply the deepest nesting of existential restrictions. Formally, the function qdepth is defined inductively for all $\mathcal{EL}$ concepts by

$$\begin{aligned}
\mathsf{qdepth}(\top) &= 0 \\
\mathsf{qdepth}(A) &= 0 \\
\mathsf{qdepth}(C \sqcap D) &= \max(\mathsf{qdepth}(C), \mathsf{qdepth}(D)) \\
\mathsf{qdepth}(\exists r.C) &= \mathsf{qdepth}(C) + 1.
\end{aligned}$$

We may also define the quantifier depth of an ELQ $q(x_0)$ as the codepth of its answer variable x_0.

Unfortunately, fitting ELQs of minimal quantifier depth can still be of large size. We show that fitting algorithms that produce ELQs of minimal quantifier depth are not sample-efficient PAC learning algorithms. To show this, we cannot use a similar construction as in the last two proofs, and use homomorphism dualities relative to some path example, as we do not use path-shaped queries. We require *unrelativized* homomorphism dualities or homomorphism dualities relative to tree-shaped examples, but there is no known construction, even for path examples, that yields such homomorphism dualities of polynomial size. Therefore, we use the weaker notion of *simulation dualities*, which suffice for ELQs. Recall that $\preceq_{\mathcal{EL}}$ denotes the existence of an $\mathcal{EL}$ simulation (Definition 5.25), and that $\mathcal{EL}$ simulations are closely linked to ELQs (see also Lemma 5.26). The proof is standard and omitted.

Lemma 6.20. *Let $q(x)$ be an ELQ and $(\mathcal{A}, a)$ an example. Then, $\mathcal{A}, \emptyset \models q(a)$ if and only if $\mathcal{A}_q, x \preceq_{\mathcal{EL}} \mathcal{A}, a$.*

This is a consequence of the semantics of CQs (and therefore also of ELQs) and the fact that the existence of a homomorphism and the existence of an $\mathcal{EL}$ simulation coincide if the *source* example is tree-shaped.

Definition 6.21 (Simulation duality)**.** Let Σ be a signature. A Σ *simulation duality* is a pair of finite sets of unary examples (F, D) such that for all examples $(\mathcal{B}, b)$ with $\mathsf{sig}(\mathcal{B}) \subseteq \Sigma$, the following are equivalent:

1. $\mathcal{A}_f, a_f \preceq_{\mathcal{EL}} \mathcal{B}, b$ for some $(\mathcal{A}_f, a_f) \in F$,
2. $\mathcal{B}, b \not\preceq_{\mathcal{EL}} \mathcal{A}_d, a_d$ for all $(\mathcal{A}_d, a_d) \in D$.

The main use of Definition 6.21 lies in the following property. Consider a Σ simulation duality (F, D) where $F = \{(\mathcal{A}_q, a_q)\}$ for some ELQ q. Then for any example $(\mathcal{B}, b)$ that uses only symbols from Σ with $\mathcal{B}, \emptyset \not\models q(b)$, there must be an $(\mathcal{A}, a) \in D$ with $\mathcal{B}, b \preceq_{\mathcal{EL}} \mathcal{A}, a$, due to Lemma 6.20. We show that for ELQs, such a set D always exists and is of polynomial size. Indeed, we show that this not only holds for ELQs, but also for examples that are directed acyclic graph shaped (DAG-shaped), since the notion of simulation dualities is of independent interest.

An ABox $\mathcal{A}$ is *DAG-shaped* if its underlying directed graph is a DAG and there are no role assertions of the form $r(x, y), s(y, x) \in \mathcal{A}$. Similar to the codepth of tree-shaped ABoxes, we define a notion of *codepth* for DAG-shaped ABoxes. The codepth of an individual a is 0 if there is no assertion $r(a, b) \in \mathcal{A}$ and the maximum of the codepths of all individuals b with $r(a, b) \in \mathcal{A}$ plus 1 otherwise.

Lemma 6.22. *Let Σ be a signature and $(\mathcal{A}, a)$ an example such that $\mathcal{A}$ is DAG-shaped and $\mathsf{sig}(\mathcal{A}) \subseteq \Sigma$. Then, we can compute in polynomial time a set D such that $(\{(\mathcal{A}, a)\}, D)$ is a Σ simulation duality. Moreover, if $\mathcal{A}$ contains exactly one Σ assertion that mentions a, then D is a singleton set.*

Proof. Let $(\mathcal{A}, a)$ be an example with $\mathcal{A}$ DAG-shaped and Σ a signature. We construct the required set D as follows. First, we define an ABox $\mathcal{A}^*$ which uses the following individuals

$$\begin{aligned}\mathsf{ind}(\mathcal{A}^*) = \{\top\} \cup \\ \{\langle b, A(b)\rangle \mid A(b) \in \mathcal{A}, A \in \Sigma\} \cup \\ \{\langle b, r(b,c)\rangle \mid r(b,c) \in \mathcal{A}, r \in \Sigma\}\end{aligned}$$

and include the following assertions, for all $\langle b, A(b)\rangle \in \mathsf{ind}(\mathcal{A}^*)$ and $\langle b, r(b,c)\rangle \in \mathsf{ind}(\mathcal{A}^*)$:

(i) $B(\top)$ for all $B \in \Sigma \cap \mathsf{N_C}$;

(ii) $s(\top, \top)$ for all $s \in \Sigma \cap \mathsf{N_R}$;

(iii) $B(\langle b, A(b)\rangle)$ for all $B \in \Sigma \cap \mathsf{N_C}$ with $B \neq A$;

(iv) $s(\langle b, A(b)\rangle, \top)$ for all $s \in \Sigma \cap \mathsf{N_R}$;

(v) $B(\langle b, r(b,c)\rangle)$ for all $B \in \Sigma \cap \mathsf{N_C}$;

(vi) $s(\langle b, r(b,c)\rangle, \top)$ for all $s \in \Sigma \cap \mathsf{N_R}$ with $s \neq r$;

(vii) $r(\langle b, r(b,c)\rangle, \langle c, \alpha\rangle)$ for all $\langle c, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)$.

We prove two auxiliary claims.

Claim 1. For all $b \in \mathsf{ind}(\mathcal{A})$ and $\langle b, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)$, $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, \alpha\rangle$.

Proof of Claim 1. We prove the claim by induction on the codepth of b in $\mathcal{A}$. If b has codepth 0, then α is of the form $A(b)$, for $A(b) \in \mathcal{A}$. By Point (iii) in the definition of $\mathcal{A}^*$, $A(\langle b, A(b)\rangle) \notin \mathcal{A}^*$, and thus $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, \alpha\rangle$.

Now, let b have codepth greater than 0. We distinguish cases on the shape of α.

- If α is of the form $A(b)$ for some $A(b) \in \mathcal{A}$, then we can argue as in the base case that $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, \alpha\rangle$.
- If α is of the form $r(b,c)$ for some $r(b,c) \in \mathcal{A}$, assume for contradiction that there is an $\mathcal{EL}$ simulation S from $\mathcal{A}$ to $\mathcal{A}^*$ with $(b, \langle b, r(b,c)\rangle) \in S$. Since S is an $\mathcal{EL}$ simulation and c is an r-successor of b in $\mathcal{A}$, there has to be an r-successor c' of $\langle b, r(b,c)\rangle$ in $\mathcal{A}^*$ with $(c, c') \in S$. By Point (vi) and (vii), c' is of shape $\langle c, \alpha\rangle$. But then $\mathcal{A}, c \preceq_{\mathcal{EL}} \mathcal{A}', \langle c, \alpha\rangle$, contradicting the induction hypothesis.

This completes the proof of Claim 1.

Claim 2. For all $b \in \mathsf{ind}(\mathcal{A})$ and examples $(\mathcal{B}, c)$ that only use symbols from Σ, if $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{B}, c$ then there is a $\langle b, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)$ such that $\mathcal{B}, c \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, \alpha\rangle$.

Proof of Claim 2. We prove the claim by induction on the codepth of b in $\mathcal{A}$. If b has codepth 0 and $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{A}', c$, then there is a concept name $A \in \Sigma$ such that $A(b) \in \mathcal{A}$ and $A(c) \notin \mathcal{B}$. It can be verified using Points (i)–(iii) above that the relation

$$S = \{(c, \langle b, A(b)\rangle)\} \cup \{(c', \top) \mid c' \in \mathsf{ind}(\mathcal{B})\}$$

is an $\mathcal{EL}$ simulation from $\mathcal{B}$ to $\mathcal{A}^*$ with $(c, \langle b, A(b)\rangle) \in S$ as required.

Now, let b have codepth greater than 0 and assume $\mathcal{A}, b \not\preceq_{\mathcal{EL}} \mathcal{B}, c$. We distinguish cases on why the latter is the case:

- If there is a concept name $A \in \Sigma$ such that $A(b) \in \mathcal{A}$ and $A(c) \notin \mathcal{B}$, we can argue as in the base case that $\mathcal{B}, c \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, A(b)\rangle$.
- If there is an assertion $r(b, b') \in \mathcal{A}$ such that for all $r(c, c') \in \mathcal{B}$, $\mathcal{A}, b' \not\preceq_{\mathcal{EL}} \mathcal{B}, c'$. We show that $\mathcal{B}, c \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle b, r(b, b')\rangle$.

  The induction hypothesis implies that for all $r(c, c') \in \mathcal{B}$ there is an $\langle b', \beta\rangle \in \mathsf{ind}(\mathcal{A}^*)$ and a simulation $S_{c'}$ from $\mathcal{B}$ to $\mathcal{A}^*$ with $(c', \langle b', \beta\rangle) \in S_{c'}$. It can be verified using Points (v)–(vii) above that

  $$S = \{(b, \langle b, r(b, b')\rangle)\} \cup \{(c', \top) \mid c' \in \mathsf{ind}(\mathcal{B})\} \cup \bigcup_{r(c,c') \in \mathcal{B}} S_{c'}$$

  is a simulation from $\mathcal{A}'$ to $\mathcal{A}^*$ with $(b, \langle b, r(b, b')\rangle) \in S$.

This completes the proof of Claim 2.

We now show that the set

$$D_a = \{(\mathcal{A}^*, \langle a, \alpha\rangle) \mid \langle a, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)\}$$

forms a Σ simulation duality together with $\{(\mathcal{A}, a)\}$.

Suppose that $\mathcal{A}, a \not\preceq_{\mathcal{EL}} (\mathcal{B}, b)$ for some example $(\mathcal{B}, b)$ with $\mathsf{sig}(\mathcal{B}) \subseteq \Sigma$. Then Claim 2 implies that there is some $\langle a, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)$ with $\mathcal{B}, b \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle a, \alpha\rangle$. It remains to note that $(\mathcal{A}^*, \langle a, \alpha\rangle) \in D_a$. Conversely, suppose that $\mathcal{A}, a \preceq_{\mathcal{EL}} \mathcal{B}, b$ and assume for showing a contradiction that $\mathcal{B}, b \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle a, \alpha\rangle$ for some $\langle a, \alpha\rangle \in \mathsf{ind}(\mathcal{A}^*)$. Since $\preceq_{\mathcal{EL}}$ is transitive, we obtain $\mathcal{A}, a \preceq_{\mathcal{EL}} \mathcal{A}^*, \langle a, \alpha\rangle$, in contradiction to Claim 1.

Clearly, D_a is a singleton set if $\mathcal{A}$ contains only a single assertion mentioning a. It remains to analyze the size of $\mathcal{A}^*$. Points (i) and (ii) together contribute $|\Sigma|$ assertions to $\mathcal{A}^*$. Points (iii) and (iv) contribute together $|\Sigma| \cdot n_C$ assertions to $\mathcal{A}^*$,

where n_C denotes the number of assertions of shape $A(b)$ in $\mathcal{A}$. Points (v) and (vi) contribute $|\Sigma| \cdot n_R$ assertions where n_R denotes the number of assertions of shape $r(b, c)$ in $\mathcal{A}$. Finally, Point (vii) contributes $|\mathcal{A}|^2$ assertions. Overall, the number of assertions in $\mathcal{A}^*$ is bounded by a polynomial in $|\Sigma|$ and $|\mathcal{A}|$. Therefore, it can be computed in polynomial time. □

Complementing Lemma 6.22, it is known that examples that are not DAG-shaped (contain a directed cycle) do not have finite simulation dualities [tCat+23c]. We find it remarkable to recall that DAG-shaped databases (that contain undirected cycles) do, in general, not have finite homomorphism dualities [NT00].

For ELQs instead of DAG-shaped examples, a result similar to Lemma 6.22 was obtained by Fortin et al. [For+22, Theorem 30]. There, dualities are called *split partners*, as the two sets F and D *split* the set of all examples into two. Fortin et al. also describe a construction of $\preceq_{\mathcal{ELI}}$ dualities of exponential size, and show that in general no polynomial size $\preceq_{\mathcal{ELI}}$ dualities exist.

Now, we continue by using Lemma 6.22 to show that minimal quantifier depth fitting algorithms are not sample-efficient PAC learning algorithms.

Theorem 6.23. *Let* **A** *be a fitting algorithm for ELQs that always produces a fitting of minimal quantifier depth. Then,* **A** *is not a sample-efficient PAC learning algorithm.*

Proof. Assume to the contrary that there is a sample-efficient PAC learning algorithm **A** that produces a most shallow fitting concept, if it exists, with associated polynomial sample size $m\colon \mathbb{R}^2 \times \mathbb{N}^4 \to \mathbb{N}$ as in Definition 3.13.

Choose $\Sigma = \{r, s, t\}$, $\delta = 0.5$, $\epsilon = 0.4$, and n large enough such that

$$\frac{2^n!}{2^{np(n)}(2^n - p(n))!} > 1 - \delta \qquad (*)$$

where $p(n)$ is the polynomial

$$p(n) = m\Big(\frac{1}{\delta}, \frac{1}{\epsilon}, 0, |\Sigma|, p_1(n), p_2(n)\Big)$$

and p_1 and p_2 are fixed polynomials that describe the size of the target query and of the examples that we are going to use respectively. Lemma 6.24 below shows that such an n always exists, regardless of the precise polynomial p. The meaning of the expression on the left-hand side of $(*)$ will become clearer later.

We use the target query

$$q_T(x_0) \leftarrow t(x_0, x_1) \wedge \cdots \wedge t(x_n, x_{n+1})$$

with quantifier depth $n+1$. We construct (both positive and negative) examples such that, with high probability, the drawn examples admit a fitting of quantifier depth n that does not generalize well. Define a set of 2^n ELQs

$$S = \{q(y_0) \leftarrow r_1(y_0, y_1) \wedge \cdots \wedge r_n(y_{n-1}, y_n) \mid r_i \in \{r, s\},\ 1 \leq i \leq n\}.$$

For each $q \in S$, we can construct by Lemma 6.22 an example (P_q, a) such that $(\{(\mathcal{A}_q, a_q)\}, \{(P_q, a)\})$ is a Σ simulation duality. By the properties of simulation dualities, (P_q, a) is a positive example for q_T. Also by Lemma 6.22, there is a single example $(\mathcal{B}_T, a)$ such that $(\{(\mathcal{A}_{q_T}, a_{q_T})\}, \{(\mathcal{B}_T, a)\})$ is a Σ simulation duality. For each $q \in S$, we construct a negatively labeled example (N_q, a) by taking

$$(N_q, a) = (P_q \times \mathcal{B}_T, (a, a)).$$

Both the positive examples and the negative examples are of size polynomial in n. We let $p_2(n)$ be any polynomial that bounds (from above) the size of the examples.

Note that by the properties of dualities and products, for all $q \in S$ and for all ELQs q' that only use symbols from Σ,

(i) $P_q, \varnothing \models q'(a)$ if and only if $q' \not\subseteq_\varnothing q$, and

(ii) $N_q, \varnothing \models q'(a)$ if and only if $q' \not\subseteq_\varnothing q$ and $q' \not\subseteq_\varnothing q_T$.

To see Point (i) note that $P_q, \varnothing \models q'(a)$ if and only if (by Lemma 6.20) $\mathcal{A}_{q'}, a_{q'} \preceq_{\mathcal{EL}} P_q, a$ if and only if (by duality) $\mathcal{A}_q, a_q \not\preceq_{\mathcal{EL}} \mathcal{A}q', a_{q'}$ if and only if (by Lemma 6.20) $q' \not\subseteq_\varnothing q$. Point (ii) can be shown similarly and uses the fact that for all unary data examples $(\mathcal{A}, a), (\mathcal{A}_1, a_1), (\mathcal{A}_2, a_2)$, $\mathcal{A}, a \preceq_{\mathcal{EL}} \mathcal{A}_1 \times \mathcal{A}_2, (a_1, a_2)$ if and only if $\mathcal{A}, a \preceq_{\mathcal{EL}} \mathcal{A}_1, a_1$ and $\mathcal{A}, a \preceq_{\mathcal{EL}} \mathcal{A}_2, a_2$, similar to Lemma 3.3.

Let P be the probability distribution that assigns probability $\frac{1}{2^{n+1}}$ to (P_q, a) and (N_q, a) for every $q \in S$, and probability 0 to all other examples. Now, assume that $\mathbf{A}$ is started on a collection of $k = m(1/\delta, 1/\epsilon, 0, |\Sigma|, p_1(n), p_2(n))$ examples E drawn from P labeled according to q_T, and then outputs a hypothesis q_H.

Note that the probability of sampling ℓ *different* objects from an N-element set is the ratio of those sequences of length ℓ that contain pairwise distinct elements in the set of all sequences of length ℓ, that is,

$$\frac{\prod_{i=0}^{\ell-1}(N-i)}{N^\ell} = \frac{N!}{N^\ell \cdot (N-\ell)!}.$$

We apply this observation to $N = 2^n$ and $\ell = k$. By choice of n, with probability $> 1 - \delta$ we have that for no $q \in S$, both $(N_q, a, -) \in E$ and $(P_q, a, +) \in E$. We show that if this is the case, then the error of q_H is strictly larger than ϵ.

Consider the ELQ $q'(y_0) = \bigwedge_{(N_p,a,-)\in E} p(y_0)$, where $\bigwedge$ denotes the joining of ELQs at the answer variable. We claim that q' fits E. Since, for all $q \in S$, $q \subseteq_\varnothing q$, Point (ii) implies that $N_q, \varnothing \not\models q(a)$. Hence, q' fits all negative examples in E. Together with our assumption that for no $q \in S$, both $(N_q, a, -) \in E$ and $(P_q, a, +) \in E$, Point (i) implies that $P_q, \varnothing \models p(a)$ for all $(N_p, a, -) \in E$ and $(P_p, a, +) \in E$. Therefore, q' fits all positive examples in E.

Since q' is a fitting of quantifier depth n and **A** finds a fitting of minimal quantifier depth, q_H must have quantifier depth at most n, which implies that $q_H \not\subseteq_\varnothing q_T$.

Consider all $q \in S$. It must be that either $q_H \subseteq_\varnothing q$ or $q_H \not\subseteq_\varnothing q$. In the first case, Point (i) implies $P_q, \varnothing \not\models q_H(a)$, hence q_H labels the (positive) example (P_q, a) incorrectly. In the second case, Point (ii) implies $N_q, \varnothing \models q_H(a)$, hence q_H labels the (negative) example (N_q, a) incorrectly. Therefore,

$$\mathsf{error}_{P,q_T,\varnothing}(q_H) \geq 0.5 > \epsilon,$$

a contradiction. □

To complete the proof of Theorem 6.23, it remains to show that there always exists a suitable n.

Lemma 6.24. *For every polynomial $p(n)$,*

$$\lim_{n\to\infty} \left(\frac{2^n!}{2^{np(n)}(2^n - p(n))!} \right) = 1.$$

Proof. As argued in the proof of Theorem 6.23, the term inside the limit is a probability, so the limit is at most 1. We start with bounding the expression inside the limit from below.

$$\begin{aligned}
\frac{2^n!}{2^{np(n)}(2^n - p(n))!} &= \frac{2^n \cdot (2^n - 1) \cdot \cdots \cdot (2^n - p(n) + 1)}{(2^n)^{p(n)}} \\
&\geq \frac{(2^n - p(n) + 1)^{p(n)}}{(2^n)^{p(n)}} \\
&= \left(1 - \frac{p(n) + 1}{2^n}\right)^{p(n)}
\end{aligned}$$

It suffices to show that the limit of the last expression is 1. In order to do so, we manipulate the expression to avoid the $p(n)$ in the exponent. Let

$$\widehat{p}(n) = 1 - \frac{p(n) + 1}{2^n}.$$

Then,

$$\begin{aligned}\lim_{n\to\infty}\left(\widehat{p}(n)^{p(n)}\right) &= \lim_{n\to\infty}\left(\exp\left(\ln\left(\widehat{p}(n)^{p(n)}\right)\right)\right)\\ &= \exp\left(\lim_{n\to\infty}\left(\ln\left(\widehat{p}(n)^{p(n)}\right)\right)\right)\\ &= \exp\left(\lim_{n\to\infty}\left(p(n)\cdot\ln(\widehat{p}(n))\right)\right).\end{aligned}$$

To determine the limit of a product where one factor $p(n)$ converges to ∞ and the other $\ln(\cdot)$ converges to 0, we apply l'Hôpital's rule. Set $f(n) = \ln(1 - \frac{p(n)+1}{2^n})$ and $g(n) = 1/p(n)$, so $\lim_{n\to\infty}\frac{f(n)}{g(n)}$ is exactly the limit we want to determine (inside the $\exp(\cdot)$). L'Hôpital's rule says that if $\lim_{n\to\infty}\frac{f'(n)}{g'(n)}$ exists, then

$$\lim_{n\to\infty}\frac{f'(n)}{g'(n)} = \lim_{n\to\infty}\frac{f(n)}{g(n)}.$$

The derivatives $f'(n)$ and $g'(n)$ of $f(n)$ and $g(n)$ are:

$$\begin{aligned}f'(n) &= \frac{\ln(2)(p(n)+1) - p'(n)}{2^n - p(n) - 1}\\ g'(n) &= \frac{-p'(n)}{q(n)} \text{ for some polynomial } q(n)\end{aligned}$$

It remains to observe that $f'(n)/g'(n)$ is an expression that has an exponential 2^n in its numerator and only polynomials everywhere else. Thus,

$$\lim_{n\to\infty}\frac{f'(n)}{g'(n)} = 0 = \lim_{n\to\infty}\frac{f(n)}{g(n)},$$

which yields $\exp(0) = 1$ as desired. □

Therefore, we cannot base a sample-efficient PAC learning algorithm on *most-general*, *most-specific*, or *quantifier depth minimal* fittings. We instead continue with an algorithm that finds fitting queries of smallest size, based on *bounded fitting*.

6.4 SAT-based PAC $\mathcal{EL}$ Concept Learner

There is general interest in software systems that learn concepts from data examples for extracting knowledge from data [Bis+23; dAma20]. In Section 2.1, we briefly described existing approaches to concept learning. However, no existing system comes with formal guarantees in the sense of PAC learning, and many interesting fitting algorithms are not sample-efficient PAC learning algorithms, as we have seen

in Section 6.3. Theorem 6.10 tells us that bounded fitting algorithms can be a good foundation for query learning systems, as they are sample-efficient PAC learning algorithms, that is, they are guaranteed to generalize from few examples. Of course, to implement a bounded fitting algorithm, we need to specify and implement a concrete size-restricted fitting algorithm. Hence, the practicality of bounded fitting depends on the computational complexity of the corresponding size-restricted fitting problem.

In contrast to the general fitting problem, the size-restricted fitting problem for CQs is not coNExpTime-complete, but closer to being tractable.

Theorem 6.25 ([GLS99])**.** *The size-restricted fitting problem for CQs under the empty ontology is Σ_2^p-complete.*

For ELQs, where the general fitting problem is ExpTime-complete, the size-restricted fitting problem is even NP-complete. The NP-hardness is a direct consequence of Lemma 6.5.

Theorem 6.26 ([Hau89])**.** *The size-restricted fitting problem for ELQs under the empty ontology is NP-complete.*

These results also hold for different size measures, such as the word encoding size, or the number of variables. The NP-hardness of size-restricted fitting for ELQs seems a bit discouraging at first, but in practice many instances of NP-complete problems can be quickly decided by leveraging the efficiency of SAT solvers. In what follows, we describe an implementation of bounded fitting for ELQs under $\mathcal{ELH}^r$ ontologies, that uses a SAT solver to decide instances of the NP-complete size-restricted fitting problem.

First, we argue that adding an $\mathcal{ELH}^r$ ontology does not increase the complexity of the size-restricted fitting problem for ELQs. Recall that in Section 5.3 we defined ELQ-universal compact models of $\mathcal{EL}^r$ ontologies (see Lemma 5.19). Similar models can be constructed for $\mathcal{ELH}^r$ ontologies, by taking role inclusions into account. For every ABox $\mathcal{A}$ and $\mathcal{ELH}^r$ ontology $\mathcal{O}$, we can compute in polynomial time an interpretation $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ that is ELQ-universal, meaning that for all ELQs q and $a \in \mathsf{ind}(\mathcal{A})$, $\mathcal{A}, \mathcal{O} \models q(a)$ if and only if $\mathcal{C}_{\mathcal{A},\mathcal{O}}, \varnothing \models q(a)$ [LTW09]. As this interpretation is finite, we can view $\mathcal{C}_{\mathcal{A},\mathcal{O}}$ as an ABox and use it as a data example.

Now, given a collection of labeled data examples E and an $\mathcal{ELH}^r$ ontology $\mathcal{O}$, we denote with $E_\mathcal{O}$ the collection obtained from E by replacing each (positive or negative) example $(\mathcal{A}, a, \cdot)$ with $(\mathcal{C}_{\mathcal{A},\mathcal{O}}, a, \cdot)$, where $\cdot \in \{+, -\}$. The following lemma shows that a fitting algorithm for ELQs under the empty ontology gives rise to a fitting algorithm for ELQs under $\mathcal{ELH}^r$ ontologies with at most a polynomial increase in running time. It is immediate from the definition of ELQ-universality.

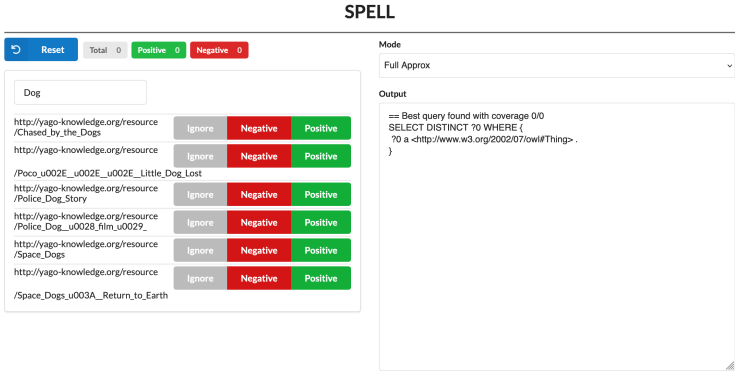

Figure 6.3: A screenshot of the demo interface of SPELL

Lemma 6.27. *An ELQ q fits a collection of labeled examples E under an $\mathcal{ELH}^r$ ontology $\mathcal{O}$ if and only if q fits $E_{\mathcal{O}}$ under the empty ontology.*

In contrast to ELQs, ELIQ-universal finite models of ontologies need not exist, as discussed in Section 5.3 (see Example 5.21). The same holds for CQ-universal models. Hence, we cannot hope to apply anything similar to Lemma 6.27 for PAC learning of ELIQs or CQs.

We implemented bounded fitting for ELQs and $\mathcal{ELH}^r$ ontologies as the system SPELL, short for **S**AT-based **P**AC $\mathcal{EL}$ concept **L**earner. SPELL is available at `https://github.com/spell-system/SPELL`, and includes a demo interface for selecting positive and negative examples interactively, displayed in Figure 6.3. On the left side, individuals from a searchable list can be selected as either positive or negative examples, and the right side is updated in real-time to show the fitting query found by SPELL. SPELL takes as input an $\mathcal{ELH}^r$ ontology $\mathcal{O}$ and an ABox $\mathcal{A}$ in OWL RDF/XML format, as well as a collection E of positive and negative examples using $\mathcal{A}$. SPELL outputs a fitting ELQ, represented as a SPARQL query. SPELL is implemented in Python 3 and uses the PySat library to interact with the SAT solver Glucose 4. For benchmarking, it provides integration into the SML-Bench framework [Wes+19].

As a first step, SPELL removes the ontology $\mathcal{O}$ by replacing the given examples E with $E_{\mathcal{O}}$ as per Lemma 6.27. It then runs bounded fitting in the variant where in each round n, the algorithm searches for a fitting ELQ with at most n variables (rather than a fitting ELQ q with $\|q\| \leq n$). The existence of such a fitting is checked using a reduction to SAT and the SAT solver. By Lemma 6.11 and the argument in the proof of Theorem 6.10, this variant of bounded fitting is also a sample-efficient

PAC learning algorithm with sample size

$$O\Big(\frac{1}{\epsilon} \cdot \log \frac{1}{\epsilon} \cdot \log \frac{1}{\delta} \cdot |\Sigma| \cdot \|q_T\|\Big).$$

We prefer this variant for implementation despite the additional factor $|\Sigma|$ in the sample complexity compared to Theorem 6.10 because it admits a more natural reduction to SAT, which we describe next.

The SAT Encoding

From $E_{\mathcal{O}}$ and the bound n, SPELL constructs a propositional formula $\varphi = \varphi_1 \wedge \varphi_2$ that is satisfiable if and only if there is an ELQ q over $\Sigma = \mathsf{sig}(E_{\mathcal{O}})$ with at most n variables that fits $E_{\mathcal{O}}$. Indeed, any model of φ returned by the SAT solver intuitively represents such an ELQ, encoded as follows. The formula φ_1 ensures that such a model represents an ELQ $q(z_1)$ as a conjunction of atoms over variables $z_1, \dots, z_n$. In φ_1, we use Boolean variables of the form $c_{i,A}$ to express that $A(z_i)$ is an element of this conjunction, and Boolean variables $x_{j,r}$ and $y_{i,j}$ to express that $r(z_i, z_j)$ occurs in this conjunction. Since the atoms should form an acyclic and rooted query, we enforce that an atom of shape $r(z_i, z_j)$ occurs exactly once for each j, and that $i < j$. The second part φ_2 then enforces that q fits $E_{\mathcal{O}}$. Let $\mathcal{A}$ be the disjoint union of all databases that occur in an example in $E_{\mathcal{O}}$. The encoding uses Boolean variables $s_{i,a}$, with $1 \leq i \leq n$ and $a \in \mathsf{ind}(\mathcal{A})$, to express that $\mathcal{A}, \emptyset \models q_{z_i}(a)$. The exact definition of φ_2 uses $\mathcal{EL}$ simulations and relies on Lemma 5.26.

We now make the formula φ precise. For encoding the ELQ q, the formula φ_1 contains the following clauses for each i with $2 \leq i \leq n$:

$$\bigvee_{j=1}^{i-1} y_{j,i} \tag{6.1}$$

$$\neg y_{j_1,i} \vee \neg y_{j_2,i} \qquad \text{for all } j_1, j_2 \text{ with } 1 \leq j_1 < j_2 < i \tag{6.2}$$

$$\bigvee_{r \in \Sigma \cap \mathsf{N_R}} x_{i,r} \tag{6.3}$$

$$\neg x_{i,r} \vee \neg x_{i,r'} \qquad \text{for all } r, r' \in \Sigma \cap \mathsf{N_R} \text{ with } r \neq r' \tag{6.4}$$

The clauses (6.1) and (6.2) ensure that z_j appears in exactly one atom of the form $r(z_i, z_j)$ and that $i < j$ for all such atoms. Clauses (6.3) and (6.4) ensure that there is a unique such role name.

The formula φ_2 makes sure that q fits $E_{\mathcal{O}}$ by enforcing that for all i with $1 \leq i \leq n$ and $a \in \mathsf{ind}(\mathcal{A})$,

$$\begin{aligned} s_{i,a} \text{ is true in a model of } \varphi \text{ if and only if } \mathcal{A}, \emptyset \models q_{z_i}(a) \\ \text{if and only if } \mathcal{A}_{q_{z_i}}, z_i \preceq_{\mathcal{EL}} \mathcal{A}, a. \end{aligned} \tag{$*$}$$

To achieve this, we express the properties of $\mathcal{EL}$ simulations (see Definition 5.25) in terms of clauses. The challenge is to capture both directions of the *if and only if* in $(*)$ efficiently, in a way that does not produce too many clauses, even for large examples.

In order to make the encoding more efficient for examples that contain many similar individuals, we use the notion of *types* that are sets of concept names. For all $a \in \mathsf{ind}(\mathcal{A})$, let $\mathsf{type}(a)$ be the set $\{A \in \mathsf{N_C} \mid A(a) \in \mathcal{A}\}$. Let TP then be the set $\{\mathsf{type}(a) \mid a \in \mathsf{ind}(\mathcal{A})\}$ of all types in that occur $\mathcal{A}$. We introduce auxiliary variables $t_{i,\tau}$, for every $1 \leq i \leq n$ and $\tau \in \mathsf{TP}$ with the intention that $t_{i,\tau}$ is true in a model if and only if the concept names of all atoms $A(z_i)$ are contained in the type τ. This is enforced by including in φ_2 the following clauses for all i with $1 \leq i \leq n$ and all types $\tau \in \mathsf{TP}$:

$$\neg t_{i,\tau} \vee \neg c_{i,A} \qquad \text{for all } A \in (\Sigma \cap \mathsf{N_C} \setminus \tau) \tag{6.5}$$

$$t_{i,\tau} \vee \bigvee_{A \in (\Sigma \cap \mathsf{N_C} \setminus \tau)} c_{i,A}. \tag{6.6}$$

The following clauses then enforce the condition that concept names are preserved by simulations, for i with $1 \leq i \leq n$ and all $a \in \mathsf{ind}(\mathcal{A})$:

$$\neg s_{i,a} \vee t_{i,\mathsf{type}(a)}. \tag{6.7}$$

This, however, only captures the *only-if*-direction of the *if and only if* in $(*)$ for concept names. To encode the other direction and the $\mathcal{EL}$ simulation condition for role names, we introduce further auxiliary variables $d_{i,j,a}$ that represent *defect*s, that is violations of the $\mathcal{EL}$ simulation conditions. More precisely, a variable $d_{i,j,a}$ is true in a model if and only if $\mathcal{A}_{q_{z_i}}, z_i \not\preceq_{\mathcal{EL}} \mathcal{A}, a$ and the variable z_j is an r-successor of z_i that is not simulated in any r-successor of a. Here, the role name r is uniquely determined by j by the Clauses (6.3) and (6.4). This behavior of the $d_{i,j,a}$ variables is achieved by the following clauses for all i, j with $1 \leq i < j \leq n$, $r \in \Sigma \cap \mathsf{N_R}$, $a \in \mathsf{ind}(\mathcal{A})$, and all $r(a,b) \in \mathcal{A}$:

$$s_{i,a} \vee \neg t_{i,\mathsf{type}(a)} \vee \bigvee_{k=i+1}^{n} d_{i,k,a} \tag{6.8}$$

$$d_{i,j,a} \vee \neg y_{i,j} \vee \neg x_{j,r} \vee \bigvee_{r(a,c) \in \mathcal{A}} s_{j,c} \tag{6.9}$$

$$\neg s_{i,a} \vee \neg d_{i,j,a} \tag{6.10}$$

$$\neg d_{i,j,a} \vee y_{i,j} \tag{6.11}$$

$$\neg d_{i,j,a} \vee \neg x_{j,r} \vee \neg s_{j,b} \tag{6.12}$$

As an example, Clause (6.11) can be read as follows: if there is a defect $d_{i,j,a}$, then $y_{i,j}$ must be true, meaning that an atom $r(z_i, z_j)$ occurs in q for some role name r.

The fact that q fits E_O is then expressed using $(*)$ as the clauses

$$\bigwedge_{(\mathcal{A}',a,+)\in E_O} s_{1,a} \quad \wedge \quad \bigwedge_{(\mathcal{A},a,-)\in E_O} \neg s_{1,a}.$$

These clauses suffice for correctness of the encoding. Summing up, the whole formula produced by SPELL in round n uses a number of variables that is in $O(n^2 + n \cdot \|\mathcal{A}\|)$, a number of clauses that is in $O(n^3 \cdot |\Sigma| \cdot |\mathsf{ind}(\mathcal{A})|)$ and in total has size in $O(n^3 \cdot |\Sigma| \cdot \|\mathcal{A}\|)$, which is linear in $\|\mathcal{A}\|$.

We have implemented two improvements of this basic encoding in SPELL. The first improvement is based on the simple observation that for computing a fitting ELQ with n variables, for every example $(\mathcal{A}, a, \cdot) \in E_O$ it suffices to consider individuals that can be reached via at most $n-1$ role assertions from a. Moreover, we may restrict the signature Σ to contain only symbols that occur in all $n-1$-reachable parts of the positive examples.

The second improvement is based on the observation that the search space for models of φ contains significant *symmetries*, in the sense that equivalent ELQs are encoded differently.

Example 6.28. Consider the ELQs

$$q_1(z_1) \leftarrow r(z_1, z_2) \wedge s(z_2, z_3) \wedge t(z_1, z_4)$$

and

$$q_2(z_1) \leftarrow r(z_1, z_2) \wedge s(z_2, z_4) \wedge t(z_1, z_3).$$

The only difference between q_1 and q_2 are the names of existential variables, and therefore $q_1 \equiv_\varnothing q_2$. In the search of a fitting ELQ of size n, it does not make sense to consider both q_1 and q_2, considering one of the two suffices. However, they are encoded differently in models of φ

To avoid this kind of symmetry in round n of bounded fitting, we add clauses that permit for every tree-shaped graph G with n vertices only a single canonical assignment of the variable names $z_1, \dots, z_n$ to the vertices of G. In order to produce the clauses, we enumerate outside the SAT solver all possible tree-shaped graphs with n vertices. The four tree-shaped graphs with four vertices are displayed in Figure 6.4 using their canonical variable names. For each such graph G, we introduce a propositional variable x_G and encode (in a straightforward way) that x_G is true if and only if $z_1, \dots, z_n$ are assigned to the vertices of G in the canonical way. We then assert using a disjunction that one of the x_G has to be satisfied. However, note that the number of tree-shaped graphs grows exponentially [OEI24] and therefore we

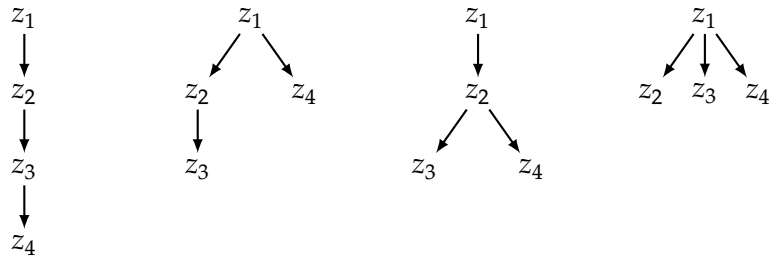

Figure 6.4: The four tree-shaped graphs with four vertices

only add these clauses if $n < 12$, to avoid spending too much time and undoing the benefit of breaking this symmetry. There are other symmetries in the search space that are not covered by these clauses.

Example 6.29. Consider the ELQs

$$q_1(z_1) \leftarrow r(z_1, z_2) \wedge r(z_1, z_3) \wedge A(z_2) \wedge B(z_3)$$

and

$$q_2(z_1) \leftarrow r(z_1, z_2) \wedge s(z_1, z_3) \wedge B(z_2) \wedge A(z_3).$$

Again, $q_1 \equiv_\varnothing q_2$, but q_1 and q_2 are represented by different models of φ.

Currently, SPELL does try to break any other symmetries than the one described above, and it is unclear if doing so is necessarily beneficial for the running time of SPELL.

This completes the description of SPELL.

6.5 Performance of SPELL

In order to determine whether the bounded fitting and SAT-solving approach of SPELL is practical, we evaluate its performance on several ELQ learning benchmarks. We compare SPELL to the ELTL component of the DL-Learner system[3], as it is the only other system for learning ELQs or $\mathcal{EL}$ concepts we are aware of. ELTL is not based on bounded fitting, but on the refinement-based approach briefly described in Section 2.1.

Existing benchmarks for concept learning systems are often aimed at learning concepts from the $\mathcal{ALC}$ family of description logics, and fitting ELQs seldom

[3]The version of DL-Learner we used is available at https://github.com/SmartDataAnalytics/DL-Learner as commit a7cd4441e52b6e54aefdea33a4914e9132ebfd97

exist. For example, in the popular Structured Machine Learning Benchmark (SML-Bench) [Wes+19], SPELL and ELTL both identify the concept $\top$ as the best fitting ELQs in six of the eight available benchmarks. The concept $\top$ is not a particularly useful fitting concept, since it classifies all negative examples incorrectly.

Therefore, we designed several new benchmarks based on existing knowledge bases, making sure that fitting ELQs always exist. The benchmarks and detailed instructions on how to reproduce the reported results in this section are available at `https://github.com/spell-system/benchmarks`. We believe that these benchmarks can be useful for future experimental evaluations of ELQ or $\mathcal{EL}$ concept learning systems. Next, we describe the four different types of benchmarks and report the performance[4] of SPELL and ELTL. For increased readability, we write ELQs as $\mathcal{EL}$ concepts.

YAGO Benchmarks

Our first set of benchmarks is based on the YAGO 4 knowledge base, which combines the concept and role names of schema.org with data from Wikidata [TWS20]. The ontology part of YAGO 4 consists of concept name inclusions $A \sqsubseteq B$, as well as domain and range restrictions. We used the smallest version of YAGO 4, containing only individuals that have an English Wikipedia entry. This version still contains over 40 million assertions. To speed up processing and the generation of the benchmarks, we first extracted a fragment of 12 million assertions that focuses on famous people and movies[5]. From this fragment, we generated a series of benchmarks of varying difficulty by selecting positive and negative examples as follows.

First, we selected target queries. For n with $4 \leq n \leq 9$ we used

$$C_n = \exists \mathsf{actor}. \prod_{i=1}^{n-2} \exists r_i.\top$$

where each r_i is a role name that is a property of actors in YAGO 4, like $\mathsf{alumniOf}$, award, or spouse. Note that the $\mathcal{EL}$ concept C_n uses n variables if represented as an ELQ. Then, based on a given sample size, we selected positive examples for C_n by querying the YAGO 4 fragment with C_n, and randomly selecting elements from the answers. For selecting negative examples, we needed to select non-answers of C_n in YAGO 4. To select negative examples that are also actors, but do not have all the properties required by C_n, we queried YAGO 4 with generalizations of C_n. To ensure that we obtain enough negative examples, we constructed these generalizations

[4] The reported running times were obtained on a 2021 MacBook Pro with M1 Pro Chip and 32 GB RAM.

[5] More precisely: the extracted fragment consists of all assertions that contain only individuals that can be reached in two steps in the underlying graph of YAGO 4 from a list of ~2000 famous people.

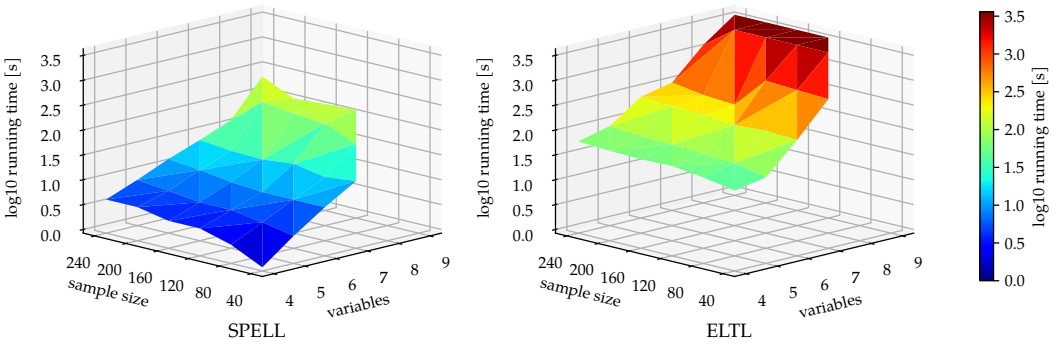

Figure 6.5: The running times of SPELL and ELTL in the YAGO benchmarks.

by first constructing the frontier of C_n (see Definition 4.13), and then dropping some immediate successors of the root in the resulting queries. The smallest fitting query of the positive and negative examples selected for C_n often has n variables as a result of this.

Example 6.30. For $n = 4$, positive examples are selected from the answers to the concept query

$$C_4 = \exists \mathsf{actor}.\big(\exists \mathsf{alumniOf}.\top \sqcap \exists \mathsf{award}.\top\big)$$

and negative examples are selected from the answers to the concept queries

$$\exists \mathsf{actor}.\exists \mathsf{alumniOf}.\top \qquad \text{and} \qquad \exists \mathsf{actor}.\exists \mathsf{award}.\top.$$

We generated benchmarks by selecting 40, 80, 120, 160, 200, and 240 examples for each target query size n with $4 \leq n \leq 9$.

The running times of SPELL and ELTL on these benchmarks, that is, the time to find a fitting concept query, are displayed in Figure 6.5. The dark red area indicates that the execution of ELTL was aborted after a timeout of 3600 s. Note that the running time axis is logarithmic. The running times of both systems behaves similarly, in that the sample size appears to have only little influence on the running time, while the number of variables of the smallest fitting query has an exponential influence on the running time. However, SPELL is approximately 1.5 orders of magnitude faster than ELTL in finding a fitting query. Specifically for $n = 8$, SPELL is able to find a fitting query in around 100 s while ELTL takes over 3600 s. This indicates in particular that SPELL can handle larger target queries.

Table 6.1: The target concepts and sample sizes of the OWL2Bench benchmarks.

	Target concept query	Sample size
o2b-1	UGC ⊓ ∃isTaughtBy.(Man ⊓ ∃likes.Music)	183
o2b-2	UGC ⊓ ∃isTaughtBy.(Woman ⊓ ∃sameHomeTown.Student)	238
o2b-3	UGC ⊓ ∃isTaughtBy.(Woman ⊓ ∃assistantProfessorOf.⊤ ⊓ ∃isCrazyAbout.⊤)	234
o2b-4	Woman ⊓ ∃teachesCourse.UGC	200
o2b-5	Woman ⊓ ∃teachesCourse.UGC ⊓ ∃dislikes.⊤	200
o2b-6	Woman ⊓ ∃teachesCourse.UGC ⊓ ∃dislikes.⊤ ⊓ ∃assistantProfessorOf.⊤	145

Table 6.2: The OWL2Bench benchmark running times in seconds, TO: > 3600s

	o2b-1	o2b-2	o2b-3	o2b-4	o2b-5	o2b-6
ELTL	TO	TO	274	580	28	152
SPELL	< 1	< 1	< 1	< 1	< 1	< 1

OWL2Bench Benchmarks

In a second set of benchmarks, we aimed to complement the first benchmarks with ones that use more features of ontologies. For this, we created six benchmarks based on the OWL2Bench ontology. Originally, OWL2Bench is a benchmark for ontology-mediated querying that combines an ABox generator with a handcrafted ontology which extends the University Ontology Benchmark [SBM20; Zho+13]. The ontology of OWL2Bench is formulated in the OWL 2 EL profile. We extracted its $\mathcal{ELH}^r$ fragment which uses all aspects of $\mathcal{ELH}^r$ and comprises 142 concept names, 83 role names, and 173 concept inclusions. We then generated a matching ABox by running the OWL2Bench generator for one university.

From this ABox and ontology we generated benchmarks using a similar process as for the YAGO knowledge base, but using six hand-designed ELQs as target queries and fixed sample sizes instead of the C_n concepts. Table 6.1 shows the target queries and the samples sizes of the six benchmarks. Again, positive examples were selected by querying with the target query and negative examples were obtained by querying with generalizations of the target query.

The running times of SPELL and ELTL on the six OWL2Bench benchmarks are displayed in Table 6.2. ELTL was aborted after a timeout of 3600 s in two cases. The difference in running times observed in the YAGO benchmarks is even more pronounced here, with SPELL returning a fitting ELQ almost instantaneously in all cases. Unfortunately, ELTL crashes immediately on these benchmarks unless the

option useMinimizer is switched off, which is enabled for all other benchmarks. We thus ran ELTL without useMinimizer, which might impact its performance.

Synthetic Benchmarks

In order to better understand the results we have observed so far, we hypothesized that SPELL and ELTL have different strengths and weaknesses based on the underlying fitting algorithms. Based on the results of the YAGO and OWL2Bench benchmarks, we anticipated that the performance of bounded fitting as implemented in SPELL would be most affected by the number of variables in the target query, whereas the performance of the refinement-based search implemented in ELTL would be most affected by the length of *specialization sequences* from $\top$, where a specialization sequence is an inverse generalization sequence (see Definition 4.33).

More formally, the *depth* of an $\mathcal{EL}$ concept query C is the length k of a sequence $C_1, \dots, C_k$ with $C_1 = \top$, $C_k = C$, and for all i,

1. $\varnothing \models C_{i+1} \sqsubseteq C_i$, and
2. for all D with $\varnothing \models C_{i+1} \sqsubseteq D$ and $\varnothing \models D \sqsubseteq C_i$, either $\varnothing \models C_{i+1} \equiv D$ or $\varnothing \models D \equiv C_i$.

Or in other words, C_{i+1} is a downward neighbor of C_i. The depth of a concept is unique and does not depend on the specific sequence [Kri18a]. It is also an orthogonal parameter to the number of variables in a query: We will see shortly that there are queries of high depth with few variables and queries of low depth with many variables.

To investigate our hypothesis, we generated synthetic benchmarks, in which we varied the number of variables and the depth of the target query systematically. Target ELQs of the first class are called *k-paths* and are of the form

$$\exists r^k.\top$$

for $k \geq 1$. These are expected to be difficult to learn for bounded fitting when the number of variables $k + 1$ becomes large, but easy to learn for refinement-based approaches as the depth of $\exists r^k.\top$ is only k. Target ELQs of the second class are called *k-1-conj* and are of the form

$$\exists r. \prod_{i=1}^{k} A_i$$

for $k \geq 1$. These have only 2 variables but depth 2^k. Target ELQs of the third class are called *k-2-conj* and are of the form

$$\exists r. \exists r. \prod_{i=1}^{k} A_i$$

Table 6.3: The running times on the synthetic benchmarks in seconds, TO: > 600s

	k-path		k-1-conj		k-2-conj	
k	ELTL	SPELL	ELTL	SPELL	ELTL	SPELL
4	1	< 1	1	< 1	1	< 1
5	1	< 1	1	< 1	4	< 1
6	1	< 1	2	< 1	394	< 1
7	1	< 1	4	< 1	TO	< 1
8	1	< 1	20	< 1	TO	< 1
9	1	< 1	124	< 1	TO	< 1
10	1	< 1	TO	< 1	TO	< 1
11	1	3	TO	< 1	TO	< 1
12	1	26	TO	< 1	TO	< 1
13	1	26	TO	< 1	TO	< 1
14	1	30	TO	< 1	TO	< 1
15	1	38	TO	< 1	TO	< 1
16	1	68	TO	< 1	TO	< 1
17	1	152	TO	< 1	TO	< 1
18	1	TO	TO	< 1	TO	< 1

for $k \geq 1$ and even have depth 2^{2^k} [Kri21]. The last two classes should be difficult to learn for refinement-based algorithms when k gets large, but easy for SPELL due to the low number of variables. For each of these classes and for all k with $4 \leq k \leq 18$, we generate a benchmark consisting of a single positive and a single negative example, which are the canonical ABoxes of the target ELQ and the single element of its frontier under the empty ontology.

Table 6.3 shows the running times of SPELL and ELTL on these benchmarks. ELTL quickly finds all k-paths, but the running time also increases quickly with increasing k on the k-1-conj and k-2-conj benchmarks. For SPELL, the situation is reversed, the k-1-conj and k-2-conj benchmarks are solved quickly, but its running time increases on the longer k-paths benchmarks, due to the size of the SAT encoding.

Generalization Benchmarks

In Section 6.4 we have shown that SPELL is a sample-efficient PAC learning algorithm. In an initial set of experiments, we wanted to see if this theoretical guarantee means that in practical benchmarks, SPELL is better able to generalize from a sample than ELTL. For this, we again used the YAGO 4 knowledge base, the target query C_6 constructed as described above, and defined a uniform probability distribution

Table 6.4: Median accuracies on the generalization benchmarks

	Accuracy	
Sample size	ELTL	SPELL
5	0.77	0.80
10	0.78	0.81
15	0.85	0.84
20	0.85	0.85
25	0.86	0.86
30	0.89	0.86
35	0.90	0.89
40	0.96	0.97
45	0.96	0.98
50	0.96	0.98
55	0.96	0.98
60	0.98	0.98
65	0.98	0.98
70	0.98	0.98
75	0.98	0.98

over all answers to the ELQ $\exists$actor.$\top$ in the YAGO 4 knowledge base. For each k with $1 \leq k \leq 15$ we then generated 20 benchmarks by independently drawing $k \cdot 5$ examples from the distribution and labeling them according to the target query. Instead of running time, we measured the accuracy of the queries that ELTL and SPELL return, with regard to the probability distribution. The median accuracies of the 20 results for each sample size are listed in Table 6.4.

As expected, fitting queries returned by SPELL generalize well to unseen examples, even when the number of training examples is small. To our surprise, ELTL exhibits the same characteristics. This may be because some heuristics of ELTL prefer fittings of smaller size, which might make ELTL an Occam algorithm and thus a sample-efficient PAC learning algorithm. It would be interesting to carry out more extensive experiments on this aspect, to determine if ELTL and SPELL have similar ability to generalize in all cases, or not.

6.6 Discussion

In this chapter, we investigated the PAC learnability of queries and presented SPELL, a sample-efficient PAC learning algorithm for ELQs under $\mathcal{ELH}^r$ ontologies. The

PAC learnability results can be summarized as follows.

- ELQs, ELIQs and CQs are not polynomial time PAC learnable and therefore also not polynomial time learnable using only equivalence queries, unless NP = RP (Theorem 6.7).
- Bounded fitting is a sample-efficient PAC learning algorithm for all query classes and all ontology languages (Theorem 6.10).
- Other interesting fitting algorithms, that produce most-general, most-specific or minimum quantifier depth fitting are not sample-efficient PAC learners (Theorems 6.18, 6.19 and 6.23).

Most importantly, the first result implies that for the polynomial time learnability results in Chapter 5, both membership and equivalence queries are necessary.

It is interesting to note that while the sample complexity of bounded fitting does not depend on the query class or ontology language (just the encoding of queries into an alphabet), its running time depends on both, as Theorem 6.26 and Theorem 6.25 demonstrate. For SPELL, we chose the advantageous combination of ELQs and $\mathcal{ELH}^r$ ontologies, to which we can apply Lemma 6.27 to remove the ontology.

Polynomial Query Learnability of ELQs. ELQs are not polynomial time learnable using only equivalence queries unless P = NP [FJL21a], but we do not know whether ELQs are polynomial query learnable using only equivalence queries. We conjecture that ELQs are polynomial query learnable using only equivalence queries, and that similar techniques as those in the proof of Theorem 6.10 can be used to show this. A result to the contrary would be surprising, due to the strong connection of PAC learning and exact learning.

On Dualities, Frontiers, and Unique Characterizations. In Section 6.3 we used *duality constructions* to obtain data examples with certain properties. It is interesting to observe that, in the setting without ontologies, dualities, frontiers and unique characterizations are closely connected [tCD22]. The connection between frontiers and unique characterizations was already discussed in Chapter 4. If $q(\overline{x})$ is a CQ, and $(\{(\mathcal{A}_q, \overline{x})\}, F)$ is a homomorphism duality relative to q, then taking $(\mathcal{A}_q, \overline{x})$ as a positive example, and F as negative examples, results in a unique characterization of q. To see this, let p be a CQ that fits these examples. Since $(\mathcal{A}_q, \overline{x})$ is a positive example, it must be that $q \subseteq_\varnothing p$. Since all elements of F are negative examples, it follows from Definition 6.14 that $p \subseteq_\varnothing q$. Similarly, it can be shown that the set $\{(\mathcal{A}_q \times \mathcal{B}, \overline{x} \otimes \overline{b}) \mid (\mathcal{B}, \overline{b}) \in F\}$ is a CQ-frontier of q. Analogous connections can be

shown for $\mathcal{EL}$ simulation dualities, ELQs and ELQ-frontiers. This connection could perhaps be used in learning algorithms. However, we are not aware of any results on the existence of dualities under ontologies.

Future Directions for SPELL

While many practical queries can be expressed as ELQs (or $\mathcal{EL}$ concepts), there are some non-relational features that often occur in concept learning benchmarks like SML-Bench, but that are not yet supported by SPELL. These are *nominals*, allowing queries to reference specific individual names in the ABox, and what OWL calls *datatype properties*, allowing individuals to possess, for example, numerical properties like age. Therefore, in order to make SPELL more useful in practice, it could be extended to learn queries that contain nominals and datatype properties. This requires efficient encoding of the numerical properties into SAT. Perhaps some inspiration for such an encoding can be taken from the techniques to learn datatype properties implemented in DL-Learner [BLW16].

It could also provide useful to investigate settings in which input examples may be labeled erroneously or according to a target query formulated in a different language than the query to be learned. In both cases, one has to admit non-perfect fittings, for which the optimization features of SAT solvers and Max-SAT solvers seem promising for efficient implementation.

Additionally, more experiments are needed to better compare the ability of SPELL to generalize from examples to other query and concept learning systems.

As is, the support of ontology languages in SPELL is limited by Lemma 6.27, since the current implementation relies on the construction of compact universal models of ontologies. SPELL could be modified to use initial segments of (infinite) universal models $\mathcal{U}_{\mathcal{A},\mathcal{O}}$ instead, to correctly handle more query classes and ontology languages. These initial segments are up to exponential in size and therefore would result in SAT encodings of exponential size. This, in general, is unavoidable as the size-restricted fitting problem for ELIQs under $\mathcal{ELI}$ ontologies is ExpTime-complete [tCat+23c], but it might not be an issue for ontologies that are used in practice, due to the small size of their universal model.

With this modification, or under the empty ontology, the SAT encoding used by SPELL could be extended to learn ELIQs, $\mathcal{ALC}$ concepts, or CQs of bounded treewidth (since the size-restricted fitting problem for CQs of bounded treewidth is in NP). Extension to all CQs is not directly possible, as the size-restricted fitting problem for CQs is Σ_2^p complete, as mentioned in Theorem 6.25. Hence, an ASP solver might be more suitable to implement bounded fitting for CQs.

Chapter 7

Conclusion

Algorithms that learn query or concept from data examples under DL ontologies can support various query and ontology engineering tasks. To be usable in practice, learning algorithms have to be *efficient* in some sense, either by running in polynomial time, or by only requiring a polynomial number of examples. In this thesis, we formalized this notion of learning algorithm for queries under ontologies in Angluin's *exact learning* model and Valiant's *PAC learning* model, and aimed to determine which query classes are efficiently learnable under which ontology languages.

This thesis is the first investigation into this question, but builds upon work in the related areas of learning queries without ontologies, learning DL concepts, and learning ontologies.

We focused on ontologies written in DLs of the $\mathcal{EL}$ and *DL-Lite* families, and on the query classes of CQs, ELIQs (acyclic and rooted unary CQs) and ELQs (tree-shaped and rooted unary CQs). Other than efficiency, we were also interested in the kinds of exact learning queries that are necessary to learn a given combination of query class and ontology language. As mentioned before, ELIQs and ELQs can also be viewed as $\mathcal{ELI}$ concepts and $\mathcal{EL}$ concepts, respectively. This means that our results also apply to the learning of concepts under ontology.

In Chapter 4, we looked at exact learning algorithms that only use membership queries. The existing approach to learn ELIQs under the empty ontology can be extended to show that ELIQs are polynomial time learnable under *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ ontologies using only membership queries. This requires significant groundwork, namely the existence of *frontiers* of ELIQs under ontologies and a bound on the length of *generalization sequences* under ontologies. Additionally, we showed that polynomial time learning with only membership queries fails for many extensions of *DL-Lite*$_{\text{core}}^{\mathcal{HF}-}$ and most importantly if ontologies contain conjunctions.

Therefore, in Chapter 5, we considered learning algorithms that use both membership queries and equivalence queries, and discussed how the algorithms can use the counterexamples provided by equivalence queries to make progress. Here, extending the known learning algorithm for CQs under the empty ontology that uses membership queries and equivalence queries to the case with ontologies is

difficult. This is mainly due to existential restrictions in the ontology that imply the existence of anonymous individuals. We first showed that ELIQs are polynomial time learnable under *DL-Lite*$_{\text{core}}^{\mathcal{HF}_{\bot}}$ ontologies using *guided generalizations* to update hypotheses with counterexamples. Then, we moved on to $\mathcal{EL}^r$ ontologies and used *compact models* to show that chordal and symmetry-free CQs, as well as symmetry-free ELIQs and ELQs are polynomial time learnable. This required generalizing several of our techniques for *disconnected CQs*. We also argued that even simple queries are not polynomial time learnable under $\mathcal{ELI}$ ontologies, and that $\mathcal{ELU}$ queries are likely also not polynomial time learnable.

In Chapter 6 we considered PAC learning of queries as it is closely related to learning with only equivalence queries. We first reviewed that (with or without ontologies) already simple query classes are not polynomial time PAC learnable, and therefore focussed on *sample-efficient* PAC learning. We showed that while several fitting algorithms are not sample-efficient, bounded fitting is sample-efficient, using a classic argument connecting PAC learning with Occam algorithms. We then presented SPELL, an implementation of bounded fitting for ELQs and $\mathcal{ELH}^r$ ontologies, and compared it to an existing implementation of an ELQ learner, with generally favorable results.

Thus, our results successfully begin to provide answers to the main question

Which query classes are efficiently learnable under which ontology languages?

However, the answers are not yet complete, as major questions remain open:

1. Are ELIQs polynomial time learnable under $\mathcal{EL}$ ontologies?
2. Are CQs polynomial time learnable under *DL-Lite* ontologies?

The techniques we used formulate learning algorithms in this thesis fail to answer these questions. This is mainly because the task of updating hypotheses with counterexamples from equivalence queries is difficult to tackle and must deal with complicated interactions of query and ontology. Answering these questions negatively is also especially difficult. No proof in the style of Theorem 5.50, or reduction from another hard learning problem, has been found yet. Answering these questions either positively or negatively will require the development of new techniques that give new insight into the behavior of queries under ontologies.

So far, we have limited the investigation into the learnability of queries under ontologies to DL ontologies and CQs that use unary and binary relations only. Learning queries that use relations of higher arity is a natural step forward. Under the empty ontology, the results concerning CQ learning hold for relations of any arity [tCD22]. However, DL ontologies can only express knowledge about unary

and binary relations. So, to drop this restriction, it becomes necessary to adopt a different family of ontology languages that can use higher-arity relations. One possible choice are *existential rules* (also known as *tuple generating dependencies*). Taking this step would raise many new questions about the learnability of queries under ontologies. It is not at all clear whether the results of this thesis transfer to the higher-arity case.

We look forward to the future developments in this research area and the insights it may provide into queries, ontologies and knowledge representation. We hope that it can serve a basis for exciting applications.

Bibliography

[ABS90] Brian Alspach, Jean-Claude Bermond, and Dominique Sotteau. "Decomposition into Cycles I: Hamilton Decompositions." In: *Cycles and Rays*. Edited by Geňa Hahn, Gert Sabidussi, and Robert E. Woodrow. Dordrecht: Kluwer Academic Publishers, 1990, pages 9–18. DOI: 10.1007/978-94-009-0517-7_2 (cited on page 87).

[Ang87] Dana Angluin. "Learning Regular Sets from Queries and Counterexamples." In: *Information and Computation* 75.2 (1987), pages 87–106. ISSN: 08905401. DOI: 10.1016/0890-5401(87)90052-6 (cited on pages 5, 36, 42).

[Ang88a] Dana Angluin. "Learning with Hints." In: *Proceedings of the First Annual Workshop on Computational Learning Theory*. COLT '88. San Francisco, CA, USA: Morgan Kaufmann, 1988, pages 167–181. URL: https://dl.acm.org/doi/10.5555/93025.93075 (visited on 05/14/2024) (cited on pages 40, 42).

[Ang88b] Dana Angluin. "Queries and Concept Learning." In: *Machine Learning* 2.4 (1988), pages 319–342. DOI: 10.1007/bf00116828 (cited on pages 36, 38, 39, 42, 162).

[Ang90] Dana Angluin. "Negative Results for Equivalence Queries." In: *Machine Learning* 5 (1990), pages 121–150. DOI: 10.1007/BF00116034 (cited on page 37).

[Ang04] Dana Angluin. "Queries Revisited." In: *Theoretical Computer Science* 313.2 (2004), pages 175–194. DOI: 10.1016/j.tcs.2003.11.004 (cited on page 40).

[AFP92] Dana Angluin, Michael Frazier, and Leonard Pitt. "Learning Conjunctions of Horn Clauses." In: *Machine Learning* 9.2–3 (1992), pages 147–164. DOI: 10.1007/bf00992675 (cited on pages 16, 162).

[AK95] Dana Angluin and Michael Kharitonov. "When Won't Membership Queries Help?" In: *Journal of Computer and System Sciences* 50.2 (1995), pages 336–355. DOI: 10.1006/jcss.1995.1026 (cited on pages 42, 163).

[AB92] Martin Anthony and Norman Biggs. *Computational Learning Theory*. Cambridge University Press, 1992 (cited on page 170).

[ADK16] Marcelo Arenas, Gonzalo I. Diaz, and Egor V. Kostylev. "Reverse Engineering SPARQL Queries." In: *Proceedings of the 25th International Conference on World Wide Web*. WWW 2016. Edited by Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao. IW3C3, 2016, pages 239–249. DOI: 10.1145/2872427.2882989 (cited on page 13).

[AK02] Marta Arias and Roni Khardon. "Learning Closed Horn Expressions." In: *Information and Computation* 178.1 (2002), pages 214–240. DOI: 10.1006/INCO.2002.3162 (cited on page 16).

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4 (cited on page 171).

[Art+09] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. "The DL-Lite Family and Relations." In: *Journal of Artificial Intelligence Research* 36 (2009), pages 1–69. DOI: 10.1613/jair.2820 (cited on pages 22, 23, 29).

[Baa03] Franz Baader. "Computing the Least Common Subsumer in the Description Logic EL w.r.t. Terminological Cycles with Descriptive Semantics." In: *Conceptual Structures for Knowledge Creation and Communication*. 11th International Conference on Conceptual Structures. Edited by Aldo de Moor, Wilfried Lex, and Bernhard Ganter. Springer, 2003, pages 117–130. DOI: 10.1007/978-3-540-45091-7_8 (cited on page 13).

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the EL Envelope." In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*. IJCAI 2005. Edited by Leslie Pack Kaelbling and Alessandro Saffiotti. San Francisco, CA, USA: Morgan Kaufmann, 2005, pages 364–369. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf (visited on 05/14/2024) (cited on page 165).

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the EL Envelope Further." In: *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*. Edited by Kendall Clark and Peter Patel-Schneider. Volume 496. CEUR Workshop Proceedings. CEUR-WS.org, 2008. URL: http://ceur-ws.org/Vol-496/owled2008dc_paper_3.pdf (visited on 05/14/2024) (cited on pages 29, 153).

[Baa+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003. ISBN: 0-521-78176-0 (cited on page 4).

[BD08] Franz Baader and Felix Distel. "A Finite Basis for the Set of EL-implications Holding in a Finite Model." In: *Formal Concept Analysis, 6th International Conference*. ICFCA 2008. Edited by Raoul Medina and Sergei A. Obiedkov. Volume 4933. Lecture Notes in Computer Science. Springer, 2008, pages 46–61. DOI: 10.1007/978-3-540-78137-0_4 (cited on page 19).

[BD09] Franz Baader and Felix Distel. "Exploring Finite Models in the Description Logic ELgfp." In: *Formal Concept Analysis, 7th International Conference*. ICFCA 2009. Edited by Sébastien Ferré and Sebastian Rudolph. Volume 5548. Lecture Notes in Computer Science. Springer, 2009, pages 146–161. DOI: 10.1007/978-3-642-01815-2_12 (cited on page 19).

[Baa+17] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. ISBN: 978-1-139-02535-5. DOI: 10.1017/9781139025355 (cited on pages 4, 31, 54).

[BKM99] Franz Baader, Ralf Küsters, and Ralf Molitor. "Computing Least Common Subsumers in Description Logics with Existential Restrictions." In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. IJCAI 99. Edited by Thomas Dean. San Francisco, CA, USA: Morgan Kaufmann, 1999, pages 96–103. URL: http://ijcai.org/Proceedings/99-1/Papers/015.pdf (visited on 05/14/2024) (cited on page 13).

[BST07] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. "Computing the Least Common Subsumer w.r.t. a Background Terminology." In: *Journal of Applied Logic* 5.3 (2007), pages 392–420. DOI: 10.1016/J.JAL.2006.03.002 (cited on pages 13, 103).

[BN00] Liviu Badea and Shan-Hwei Nienhuys-Cheng. "A Refinement Operator for Description Logics." In: *Inductive Logic Programming, 10th International Conference*. ILP 2000. Edited by James Cussens and Alan M. Frisch. Volume 1866. Lecture Notes in Computer Science. London: Springer, 2000, pages 40–59. DOI: 10.1007/3-540-44960-4_3 (cited on pages 14, 15, 34).

[Bai+12] Samantha Bail, Sandra Alkiviadous, Bijan Parsia, David Workman, Mark van Harmelen, Rafael S. Gonçalves, and Cristina Garilao. "Fish-Mark: A Linked Data Application Benchmark." In: *Proceedings of the Joint Workshop on Scalable and High-Performance Semantic Web Systems*. SSWS+HPCSW 2012. Edited by Achille Fokoue, Thorsten Liebig, Eric L. Goodman, Jesse Weaver, Jacopo Urbani, and David Mizell. Volume 943. CEUR Workshop Proceedings. CEUR-WS.org, 2012, pages 1–15. URL: https://ceur-ws.org/Vol-943/SSWS_HPCSW2012_paper1.pdf (visited on 05/14/2024) (cited on page 119).

[BR17] Pablo Barceló and Miguel Romero. "The Complexity of Reverse Engineering Problems for Conjunctive Queries." In: *20th International Conference on Database Theory*. ICDT 2017. Edited by Michael Benedikt and Giorgio Orsi. Volume 68. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 7:1–7:17. DOI: 10.4230/LIPIcs.ICDT.2017.7 (cited on pages 13, 34).

[BO15] Meghyn Bienvenu and Magdalena Ortiz. "Ontology-Mediated Query Answering with Data-Tractable Description Logics." In: *Reasoning Web. Web Logic Rules - 11th International Summer School 2015*. Edited by Wolfgang Faber and Adrian Paschke. Volume 9203. Lecture Notes in Computer Science. Springer, 2015, pages 218–307. DOI: 10.1007/978-3-319-21768-0_9 (cited on pages 4, 33).

[Bie+13] Meghyn Bienvenu, Magdalena Ortiz, Mantas Simkus, and Guohui Xiao. "Tractable Queries for Lightweight Description Logics." In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI 2013. Edited by Francesca Rossi. Menlo Park, California: AAAI Press/International Joint Conferences on Artificial Intelligence, 2013, pages 768–774. URL: https://www.ijcai.org/Proceedings/13/Papers/120.pdf (visited on 05/14/2024) (cited on pages 30, 63, 113).

[Bis+23] Tomás Bisták, Peter Svec, Ján Kluka, Alexander Simko, Stefan Balogh, and Martin Homola. "Improving DL-learner on a Malware Detection Use Case." In: *Proceedings of the 36th International Workshop on Description Logics*. DL 2023. Edited by Oliver Kutz, Carsten Lutz, and Ana Ozaki. Volume 3515. CEUR Workshop Proceedings. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3515/paper-6.pdf (cited on page 197).

[Blu94] Avrim L. Blum. "Separating Distribution-Free and Mistake-Bound Learning Models over the Boolean Domain." In: *SIAM Journal on*

Computing 23.5 (1994), pages 990–1000. DOI: 10.1137/S0097539792233455X (cited on page 43).

[Blu+23] Sophie Blum, Raoul Koudijs, Ana Ozaki, and Samia Touileb. *Learning Horn Envelopes via Queries from Large Language Models*. 2023. DOI: 10.48550/ARXIV.2305.12143. arXiv: 2305.12143. Pre-published (cited on page 6).

[Blu+89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. "Learnability and the Vapnik-Chervonenkis Dimension." In: *Journal of the ACM* 36.4 (1989), pages 929–965. DOI: 10.1145/76359.76371 (cited on pages 176–178).

[BP92] Raymond A. Board and Leonard Pitt. "On the Necessity of Occam Algorithms." In: *Theoretical Computer Science* 100.1 (1992), pages 157–184. DOI: 10.1016/0304-3975(92)90367-O (cited on page 181).

[BCL15] Angela Bonifati, Radu Ciucanu, and Aurélien Lemay. "Learning Path Queries on Graph Databases." In: *Proceedings of the 18th International Conference on Extending Database Technology*. EDBT 2015. Edited by Gustavo Alonso, Floris Geerts, Lucian Popa, Pablo Barceló, Jens Teubner, Martín Ugarte, Jan Van den Bussche, and Jan Paredaens. OpenProceedings.org, 2015, pages 109–120. DOI: 10.5441/002/edbt.2015.11 (cited on page 13).

[BDK16] Daniel Borchmann, Felix Distel, and Francesco Kriegel. "Axiomatisation of General Concept Inclusions from Finite Interpretations." In: *Journal of Applied Non-Classical Logics* 26.1 (2016), pages 1–46. DOI: 10.1080/11663081.2016.1168230 (cited on page 19).

[Bor+89] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. "CLASSIC: A Structural Data Model for Objects." In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. Edited by James Clifford, Bruce G. Lindsay, and David Maier. ACM, 1989, pages 58–67. DOI: 10.1145/67544.66932 (cited on page 17).

[Bot+16] Elena Botoeva, Roman Kontchakov, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. "Games for Query Inseparability of Description Logic Knowledge Bases." In: *Artificial Intelligence* 234 (2016), pages 78–119. DOI: 10.1016/J.ARTINT.2016.01.010 (cited on pages 32, 107).

[BS85] Ronald J. Brachman and James G. Schmolze. "An Overview of the KL-ONE Knowledge Representation System." In: *Cognitive Science* 9.2 (1985), pages 171–216. DOI: 10.1207/S15516709COG0902_1 (cited on page 2).

[Bra04] Sebastian Brandt. "Polynomial Time Reasoning in a Description Logic with Existential Restrictions, GCI Axioms, and - What Else?" In: *Proceedings of the 16th Eureopean Conference on Artificial Intelligence*. ECAI 2004. Edited by Ramó López de Mántaras and Lorenza Saitta. Amsterdam, Netherlands: IOS Press, 2004, pages 298–302. DOI: 10.5555/3000001.3000065 (cited on page 29).

[BLW16] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. "DL-Learner - A Framework for Inductive Learning on the Semantic Web." In: *Journal of Web Semantics* 39 (2016), pages 15–24. DOI: 10.1016/j.websem.2016.06.001 (cited on pages 3, 15, 211).

[Cal+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "DL-Lite: Tractable Description Logics for Ontologies." In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*. AAAI 2005. Edited by Manuela M. Veloso and Subbarao Kambhampati. AAAI Press, 2005, pages 602–607. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-094.php (visited on 05/14/2024) (cited on page 30).

[Cal+07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family." In: *Journal of Automated Reasoning* 39.3 (2007), pages 385–429. DOI: 10.1007/s10817-007-9078-x (cited on page 4).

[Cal+13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Data Complexity of Query Answering in Description Logics." In: *Artificial Intelligence* 195 (2013), pages 335–360. DOI: 10.1016/J.ARTINT.2012.10.003 (cited on page 4).

[CGL98] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. "On the Decidability of Query Containment under Constraints." In: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '98. Edited by Alberto O. Mendelzon and Jan Paredaens. ACM, 1998, pages 149–158. DOI: 10.1145/275487.275504 (cited on page 4).

[tCDK13] Balder ten Cate, Víctor Dalmau, and Phokion G. Kolaitis. "Learning Schema Mappings." In: *ACM Transactions on Database Systems* 38.4 (2013), 28:1–28:31. DOI: 10.1145/2539032.2539035 (cited on pages 6, 15, 16, 47, 101, 102, 161).

[tCD15] Balder ten Cate and Víctor Dalmau. "The Product Homomorphism Problem and Applications." In: *18th International Conference on Database Theory*. ICDT 2015. Edited by Marcelo Arenas and Martín Ugarte. Volume 31. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015, pages 161–176. DOI: 10.4230/LIPIcs.ICDT.2015.161 (cited on pages 13, 34).

[tCat+18] Balder ten Cate, Phokion G. Kolaitis, Kun Qian, and Wang-Chiew Tan. "Active Learning of GAV Schema Mappings." In: *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS '18. Edited by Jan Van den Bussche and Marcelo Arenas. ACM, 2018, pages 355–368. DOI: 10.1145/3196959.3196974 (cited on page 16).

[tCD22] Balder ten Cate and Víctor Dalmau. "Conjunctive Queries: Unique Characterizations and Exact Learnability." In: *ACM Transactions on Database Systems* 47.4 (2022), 14:1–14:41. DOI: 10.1145/3559756 (cited on pages 6, 16, 36, 47, 60, 88, 97, 183, 210, 214).

[tCK23] Balder ten Cate and Raoul Koudijs. *Characterising Modal Formulas with Examples*. 2023. DOI: 10.48550/ARXIV.2304.06646. arXiv: 2304.06646. Pre-published (cited on page 16).

[tCat+23a] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. *Extremal Fitting CQs Do Not Generalize*. Version 1. 2023. DOI: 10.48550/arXiv.2312.03407. arXiv: 2312.03407 [cs]. Pre-published (cited on page 170).

[tCat+23b] Balder ten Cate, Víctor Dalmau, Maurice Funk, and Carsten Lutz. "Extremal Fitting Problems for Conjunctive Queries." In: *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS 2023. Edited by Floris Geerts, Hung Q. Ngo, and Stavros Sintos. ACM, 2023, pages 89–98. DOI: 10.1145/3584372.3588655 (cited on pages 181–183).

[tCat+23c] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "SAT-based PAC Learning of Description Logic Concepts." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI 2023. Edited by Edith Elkind. International Joint

Conferences on Artificial Intelligence, 2023, pages 3347–3355. DOI: 10.24963/ijcai.2023/373 (cited on pages 170, 181, 194, 211).

[tCat+24] Balder ten Cate, Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "On the Non-Efficient PAC Learnability of Conjunctive Queries." In: *Information Processing Letters* 183.106431 (2024). DOI: 10.1016/J.IPL.2023.106431 (cited on pages 35, 170).

[CM77] Ashok K. Chandra and Philip M. Merlin. "Optimal Implementation of Conjunctive Queries in Relational Data Bases." In: *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*. STOC '77. Edited by John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison. ACM, 1977, pages 77–90. DOI: 10.1145/800105.803397 (cited on pages 25, 30).

[CF20] Hunter Chase and James Freitag. "Bounds in Query Learning." In: *Conference on Learning Theory*. COLT 2020. Edited by Jacob D. Abernethy and Shivani Agarwal. Volume 125. Proceedings of Machine Learning Research. PMLR, 2020, pages 1142–1160. URL: http://proceedings.mlr.press/v125/chase20a.html (visited on 06/15/2024) (cited on page 40).

[CCL21] Gianluca Cima, Federico Croce, and Maurizio Lenzerini. "Query Definability and Its Approximations in Ontology-Based Data Management." In: *The 30th ACM International Conference on Information and Knowledge Management*. CIKM '21. Edited by Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong. ACM, 2021, pages 271–280. DOI: 10.1145/3459637.3482466 (cited on pages 14, 35).

[CS01] Edmund M. Clarke and Bernd-Holger Schlingloff. "Model Checking." In: *Handbook of Automated Reasoning*. Edited by John Alan Robinson and Andrei Voronkov. Cambridge, Massachusetts: MIT Press, 2001, pages 1635–1790. ISBN: 978-0-444-50813-3 (cited on page 122).

[CW16] Sara Cohen and Yaacov Y. Weiss. "The Complexity of Learning Tree Patterns from Example Graphs." In: *ACM Transactions on Database Systems* 41.2 (2016), 14:1–14:44. DOI: 10.1145/2890492 (cited on page 13).

[CBH92] William W. Cohen, Alexander Borgida, and Haym Hirsh. "Computing Least Common Subsumers in Description Logics." In: *Proceedings of the 10th National Conference on Artificial Intelligence*. AAAI 1992. Edited by William R. Swartout. AAAI Press, 1992, pages 754–760. URL: http:

//www.aaai.org/Library/AAAI/1992/aaai92-117.php (visited on 05/14/2024) (cited on page 13).

[CH92] William W. Cohen and Haym Hirsh. "Learnability of Description Logics." In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. ACM, 1992, pages 116–127. DOI: 10.1145/130385.130398 (cited on page 17).

[CH94a] William W. Cohen and Haym Hirsh. "Learning the CLASSIC Description Logic: Theoretical and Experimental Results." In: *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*. KR 1994. Morgan Kaufmann, 1994, pages 121–133. DOI: 10.1016/B978-1-4832-1452-8.50108-1 (cited on page 17).

[CH94b] William W. Cohen and Haym Hirsh. "The Learnability of Description Logics with Equality Constraints." In: *Machine Learning* 17.2–3 (1994), pages 169–199. DOI: 10.1007/bf00993470 (cited on page 17).

[CH95] William W. Cohen and Haym Hirsh. "Corrigendum for "Learnability of Description Logics"." In: *Proceedings of the Eighth Annual Conference on Computational Learning Theory*. COLT '95. ACM, 1995, pages 463–463. ISBN: 978-0-89791-723-0. DOI: 10.1145/225298.372773 (cited on page 17).

[dAma20] Claudia d'Amato. "Machine Learning for the Semantic Web: Lessons Learnt and next Research Directions." In: *Semantic Web* 11.1 (2020), pages 195–203. DOI: 10.3233/SW-200388 (cited on page 197).

[Dal99] Víctor Dalmau. "Boolean Formulas Are Hard to Learn for Most Gate Bases." In: *Algorithmic Learning Theory, 10th International Conference*. ALT '99. Edited by Osamu Watanabe and Takashi Yokomori. Volume 1720. Lecture Notes in Computer Science. Springer, 1999, pages 301–312. DOI: 10.1007/3-540-46769-6_25 (cited on page 163).

[DN23] Caglar Demir and Axel-Cyrille Ngonga Ngomo. "Neuro-Symbolic Class Expression Learning." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI 2023. Edited by Edith Elkind. International Joint Conferences on Artificial Intelligence, 2023, pages 3624–3632. DOI: 10.24963/IJCAI.2023/403 (cited on pages 3, 15).

[DB86] Ronald D. Dutton and Robert C. Brigham. "Computationally Efficient Bounds for the Catalan Numbers." In: *European Journal of Combinatorics* 7.3 (1986), pages 211–213. DOI: 10.1016/S0195-6698(86)80024-5 (cited on page 180).

[Eit+08] Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Simkus. "Query Answering in the Description Logic Horn-SHIQ." In: *Logics in Artificial Intelligence, 11th European Conference*. JELIA 2008. Edited by Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing. Volume 5293. Lecture Notes in Computer Science. Springer, 2008, pages 166–179. DOI: 10.1007/978-3-540-87803-2_15 (cited on pages 4, 30).

[FdAE08] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. "DL-FOIL Concept Learning in Description Logics." In: *Inductive Logic Programming, 18th International Conference*. ILP 2008. Edited by Filip Zelezný and Nada Lavrac. Volume 5194. Lecture Notes in Computer Science. Springer, 2008, pages 107–121. DOI: 10.1007/978-3-540-85928-4_12 (cited on page 15).

[Fan+18] Nicola Fanizzi, Giuseppe Rizzo, Claudia d'Amato, and Floriana Esposito. "DLFoil: Class Expression Learning Revisited." In: *European Knowledge Acquisition Workshop*. EKAW 2018. Edited by Catherine Faron-Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint. Volume 11313. Lecture Notes in Computer Science. Springer, 2018, pages 98–113. DOI: 10.1007/978-3-030-03667-6_7 (cited on pages 15, 34).

[For+22] Marie Fortin, Boris Konev, Vladislav Ryzhikov, Yury Savateev, Frank Wolter, and Michael Zakharyaschev. "Unique Characterisability and Learnability of Temporal Instance Queries." In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*. KR 2022. Edited by Gabriele Kern-Isberner, Gerhard Lakemeyer, and Thomas Meyer. International Joint Conferences on Artificial Intelligence, 2022, pages 163–173. DOI: 10.24963/kr.2022/17 (cited on pages 16, 194).

[FP96] Michael Frazier and Leonard Pitt. "CLASSIC Learning." In: *Machine Learning* 25.2–3 (1996), pages 151–193. DOI: 10.1023/A:1026443024002 (cited on page 17).

[FJL21a] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Actively Learning Concepts and Conjunctive Queries under ELr-ontologies." In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. IJCAI 2021. Edited by Zhi-Hua Zhou. International Joint Conferences on Artificial Intelligence, 2021, pages 1887–1893. DOI: 10.24963/ijcai.2021/260 (cited on pages 101, 176, 210).

[FJL21b] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Actively Learning ELI Queries under DL-Lite Ontologies." In: *Proceedings of the 34th International Workshop on Description Logics*. DL 2021. Edited by Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt. Volume 2954. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2954/paper-14.pdf (visited on 05/14/2024) (cited on page 49).

[FJL22a] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Exact Learning of ELI Queries in the Presence of DL-Lite-Horn Ontologies." In: *Proceedings of the 35th International Workshop on Description Logics*. DL 2022. Edited by Ofer Arieli, Martin Homola, Jean Christoph Jung, and Marie-Laure Mugnier. Volume 3263. CEUR Workshop Proceedings. CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3263/paper-9.pdf (visited on 05/14/2024) (cited on pages 49, 101).

[FJL22b] Maurice Funk, Jean Christoph Jung, and Carsten Lutz. "Frontiers and Exact Learning of ELI Queries under DL-Lite Ontologies." In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. IJCAI-ECAI-2022. Edited by Luc De Raedt. International Joint Conferences on Artificial Intelligence, 2022, pages 2627–2633. DOI: 10.24963/ijcai.2022/364 (cited on page 49).

[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: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. IJCAI 2019. Edited by Sarit Kraus. International Joint Conferences on Artificial Intelligence, 2019, pages 1682–1688. DOI: 10.24963/ijcai.2019/233 (cited on pages 14, 34, 35, 153).

[Gli+08] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. "Conjunctive Query Answering for the Description Logic SHIQ." In: *Journal of Artificial Intelligence Research* 31 (2008), pages 157–204. DOI: 10.1613/jair.2372 (cited on page 4).

[GLS99] Georg Gottlob, Nicola Leone, and Francesco Scarcello. "On the Complexity of Some Inductive Logic Programming Problems." In: *New Generation Computing* 17.1 (1999), pages 53–75. DOI: 10.1007/BF03037582 (cited on page 198).

[Gui+21] Ricardo Guimarães, Ana Ozaki, Cosimo Persia, and Baris Sertkaya. "Mining EL Bases with Adaptable Role Depth." In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI 2021. AAAI Press, 2021,

pages 6367–6374. DOI: 10.1609/AAAI.V35I7.16790 (cited on page 19).

[GJS18] 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 the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI-ECAI 2018. Edited by Jérôme Lang. International Joint Conferences on Artificial Intelligence, 2018, pages 1847–1853. DOI: 10.24963/ijcai.2018/255 (cited on pages 14, 34, 35).

[Hau89] David Haussler. "Learning Conjunctive Concepts in Structural Domains." In: *Machine Learning* 4.1 (1989), pages 7–40. DOI: 10.1007/bf00114802 (cited on pages 16, 173, 198).

[Hau+91] David Haussler, Michael J. Kearns, Nick Littlestone, and Manfred K. Warmuth. "Equivalence of Models for Polynomial Learnability." In: *Information and Computation* 95.2 (1991), pages 129–161. DOI: 10.1016/0890-5401(91)90042-Z (cited on page 41).

[Heg95] Tibor Hegedüs. "Generalized Teaching Dimensions and the Query Complexity of Learning." In: *Proceedings of the Eigth Annual Conference on Computational Learning Theory*. COLT 1995. Edited by Wolfgang Maass. ACM, 1995, pages 108–117. DOI: 10.1145/225298.225311 (cited on page 40).

[HNZ96] P. Hell, J. Nešetřil, and X. Zhu. "Complexity of Tree Homomorphisms." In: *Discrete Applied Mathematics* 70.1 (1996), pages 23–36. ISSN: 0166218X. DOI: 10.1016/0166-218X(96)00099-6 (cited on page 173).

[Hel+96] Lisa Hellerstein, Krishnan Pillaipakkamnatt, Vijay Raghavan, and Dawn Wilkins. "How Many Queries Are Needed to Learn?" In: *Journal of the ACM* 43.5 (1996), pages 840–862. DOI: 10.1145/234752.234755 (cited on page 40).

[Hir00] Kouichi Hirata. "On the Hardness of Learning Acyclic Conjunctive Queries." In: *Algorithmic Learning Theory, 11th International Conference*. ALT 2000. Edited by Hiroki Arimura, Sanjay Jain, and Arun Sharma. Volume 1968. Lecture Notes in Computer Science. Springer, 2000, pages 238–251. DOI: 10.1007/3-540-40992-0_18 (cited on page 16).

[Hir05] Kouichi Hirata. "Prediction-Hardness of Acyclic Conjunctive Queries." In: *Theoretical Computer Science* 348.1 (2005), pages 84–94. DOI: 10.1016/j.tcs.2005.09.006 (cited on page 16).

[HKR10] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC, 2010. ISBN: 978-1-4200-9050-5. URL: http://www.semantic-web-book.org/ (visited on 05/14/2024) (cited on page 2).

[IPF07] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. "An Algorithm Based on Counterfactuals for Concept Learning in the Semantic Web." In: *Applied Intelligence* 26.2 (2007), pages 139–159. DOI: 10.1007/s10489-006-0011-5 (cited on page 15).

[Jun+20] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. "Logical Separability of Incomplete Data under Ontologies." In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*. KR 2020. Edited by Diego Calvanese, Esra Erdem, and Michael Thielscher. 2020, pages 517–528. DOI: 10.24963/kr.2020/52 (cited on pages 14, 34).

[Jun+22] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. "Logical Separability of Labeled Data Examples under Ontologies." In: *Artificial Intelligence* 313 (2022), page 103785. DOI: 10.1016/J.ARTINT.2022.103785 (cited on pages 5, 14).

[JLW20] Jean Christoph Jung, Carsten Lutz, and Frank Wolter. "Least General Generalizations in Description Logic: Verification and Existence." In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI 2020. AAAI Press, 2020, pages 2854–2861. DOI: 10.1609/aaai.v34i03.5675 (cited on pages 13, 103, 119, 140).

[Jun+23a] Jean Christoph Jung, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. *Temporalising Unique Characterisability and Learnability of Ontology-Mediated Queries*. 2023. DOI: 10.48550/arXiv.2306.07662. arXiv: 2306.07662. Pre-published (cited on page 16).

[Jun+23b] Jean Christoph Jung, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev. "Temporalising Unique Characterisability and Learnability of Ontology-Mediated Queries (Extended Abstract)." In: *Proceedings of the 36th International Workshop on Description Logics*. DL 2023. Edited by Oliver Kutz, Carsten Lutz, and Ana Ozaki. Volume 3515. CEUR Workshop Proceedings. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3515/abstract-13.pdf (visited on 05/14/2024) (cited on page 16).

[KV89] Michael J. Kearns and Leslie G. Valiant. "Cryptographic Limitations on Learning Boolean Formulae and Finite Automata." In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. STOC '89. Edited by David S. Johnson. ACM, 1989, pages 433–444. DOI: 10.1145/73007.73049 (cited on page 163).

[Kie93] Jörg-Uwe Kietz. "Some Lower Bounds for the Computational Complexity of Inductive Logic Programming." In: *European Conference on Machine Learning*. ECML-93. Edited by Pavel Brazdil. Volume 667. Lecture Notes in Computer Science. Springer, 1993, pages 115–123. DOI: 10.1007/3-540-56602-3_131 (cited on pages 16, 173).

[KKZ11] Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev. "On (in)Tractability of OBDA with OWL 2 QL." In: *Proceedings of the 24th International Workshop on Description Logics*. DL 2011. Edited by Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev. Volume 745. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-745/paper_7.pdf (visited on 05/14/2024) (cited on pages 30, 63).

[Kon+18] Boris Konev, Carsten Lutz, Ana Ozaki, and Frank Wolter. "Exact Learning of Lightweight Description Logic Ontologies." In: *Journal of Machine Learning Research* 18.201 (2018), pages 1–63. URL: http://jmlr.org/papers/v18/16-256.html (visited on 05/14/2024) (cited on page 18).

[KOW16] Boris Konev, Ana Ozaki, and Frank Wolter. "A Model for Learning Description Logic Ontologies Based on Exact Learning." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI 2016. Edited by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pages 1008–1015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11948 (visited on 05/14/2024) (cited on page 18).

[Kon+10] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. "The Combined Approach to Query Answering in DL-Lite." In: *Principles Of Knowledge Representation And Reasoning: Proceedings Of The Twelfth International Conference*. KR 2010. Edited by Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski. AAAI Press, 2010, pages 247–257. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1282 (visited on 05/14/2024) (cited on page 22).

[Kou22] Raoul Koudijs. "Learning Modal Formulas via Dualities." Master's thesis. Universiteit van Amsterdam, 2022. URL: https://eprints.illc.uva.nl/id/eprint/1957/1/MoL-2022-07.text.pdf (visited on 06/19/2024) (cited on page 16).

[Kri18a] Francesco Kriegel. "The Distributive, Graded Lattice of EL Concept Descriptions and Its Neighborhood Relation." In: *Proceedings of the Fourteenth International Conference on Concept Lattices and Their Applications*. CLA 2018. Edited by Dmitry I. Ignatov and Lhouari Nourine. Volume 2123. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pages 267–278. URL: http://ceur-ws.org/Vol-2123/paper22.pdf (visited on 05/14/2024) (cited on pages 15, 80, 207).

[Kri18b] Francesco Kriegel. *The Distributive, Graded Lattice of EL Concept Descriptions and Its Neighborhood Relation (Extended Version)*. Technical report LTCS-18-10. Dresden University of Technology, 2018. URL: https://lat.inf.tu-dresden.de/research/reports/2018/Kr-LTCS-18-10.pdf (visited on 05/14/2024) (cited on page 15).

[Kri19] Francesco Kriegel. "Constructing and Extending Description Logic Ontologies Using Methods of Formal Concept Analysis." PhD thesis. Dresden University of Technology, 2019. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-360998 (visited on 05/14/2024) (cited on page 15).

[Kri21] Francesco Kriegel. "Navigating the EL Subsumption Hierarchy." In: *Proceedings of the 34th International Workshop on Description Logics*. DL 2021. Edited by Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt. Volume 2954. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2954/paper-21.pdf (visited on 05/14/2024) (cited on pages 15, 95, 208).

[Kri24] Francesco Kriegel. "Efficient Axiomatization of OWL 2 EL Ontologies from Data by Means of Formal Concept Analysis." In: *Thirty-Eighth AAAI Conference on Artificial Intelligence*. AAAI 2024. Edited by Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan. AAAI Press, 2024, pages 10597–10606. DOI: 10.1609/AAAI.V38I9.28930 (cited on page 19).

[KRH13] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. "Complexities of Horn Description Logics." In: *ACM Transactions on Computational Logic* 14.1 (2013), 2:1–2:36. ISSN: 1529-3785. DOI: 10.1145/2422085.2422087 (cited on page 29).

[KM02] Ralf Küsters and Ralf Molitor. "Approximating Most Specific Concepts in Description Logics with Existential Restrictions." In: *AI Communications* 15.1 (2002), pages 47–59. URL: http://content.iospress.com/articles/ai-communications/aic253 (visited on 05/14/2024) (cited on page 13).

[Lan+15] Davide Lanti, Martín Rezk, Guohui Xiao, and Diego Calvanese. "The NPD Benchmark: Reality Check for OBDA Systems." In: *Proceedings of the 18th International Conference on Extending Database Technology*. EDBT 2015. Edited by Gustavo Alonso, Floris Geerts, Lucian Popa, Pablo Barceló, Jens Teubner, Martín Ugarte, Jan Van den Bussche, and Jan Paredaens. OpenProceedings.org, 2015, pages 617–628. DOI: 10.5441/002/EDBT.2015.62 (cited on page 119).

[Leh+14] Jens Lehmann, Nicola Fanizzi, Lorenz Bühmann, and Claudia d'Amato. "Concept Learning." In: *Perspectives on Ontology Learning*. Edited by Jens Lehmann and Johanna Völker. Volume 18. Studies on the Semantic Web. Amsterdam, Netherlands: IOS Press, 2014, pages 71–91 (cited on page 34).

[LH09] Jens Lehmann and Christoph Haase. "Ideal Downward Refinement in the EL Description Logic." In: *Inductive Logic Programming, 19th International Conference*. ILP 2009. Edited by Luc De Raedt. Volume 5989. Lecture Notes in Computer Science. Springer, 2009, pages 73–87. DOI: 10.1007/978-3-642-13840-9_8 (cited on page 15).

[LH10] Jens Lehmann and Pascal Hitzler. "Concept Learning in Description Logics Using Refinement Operators." In: *Machine Learning* 78.1–2 (2010), pages 203–250. DOI: 10.1007/s10994-009-5146-2 (cited on page 15).

[Lis12] Francesca A. Lisi. "A Formal Characterization of Concept Learning in Description Logics." In: *Proceedings of the 2012 International Workshop on Description Logics*. DL 2012. Edited by Yevgeny Kazakov, Domenico Lembo, and Frank Wolter. Volume 846. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: https://ceur-ws.org/Vol-846/paper_66.pdf (visited on 05/14/2024) (cited on page 15).

[Lis16] Francesca Alessandra Lisi. "A Model+solver Approach to Concept Learning." In: *XVth International Conference of the Italian Association for Artificial Intelligence*. AI*IA 2016. Edited by Giovanni Adorni, Stefano Cagnoni, Marco Gori, and Marco Maratea. Volume 10037. Lecture Notes in Computer Science. Springer, 2016, pages 266–279. DOI: 10.1007/978-3-319-49130-1_20 (cited on page 15).

[LP22] Carsten Lutz and Marcin Przybylko. *Efficient Answer Enumeration in Description Logics with Functional Roles - Extended Version*. Nov. 28, 2022. DOI: 10.48550/arXiv.2211.15248. arXiv: 2211.15248 [cs]. Pre-published (cited on page 33).

[Lut+13] Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. "The Combined Approach to OBDA: Taming Role Hierarchies Using Filters." In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference*. ISWC 2013. Edited by Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz. Volume 8218. Lecture Notes in Computer Science. Springer, 2013, pages 314–330. DOI: 10.1007/978-3-642-41335-3_20 (cited on page 119).

[LTW09] Carsten Lutz, David Toman, and Frank Wolter. "Conjunctive Query Answering in the Description Logic EL Using a Relational Database System." In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. IJCAI 2009. Edited by Craig Boutilier. AAAI Press/International Joint Conferences on Artificial Intelligence, 2009, pages 2070–2075. URL: http://ijcai.org/Proceedings/09/Papers/341.pdf (visited on 05/14/2024) (cited on pages 117, 198).

[LW10] Carsten Lutz and Frank Wolter. "Deciding Inseparability and Conservative Extensions in the Description Logic EL." In: *Journal of Symbolic Computation* 45.2 (2010), pages 194–228. DOI: 10.1016/J.JSC.2008.10.007 (cited on page 122).

[Mar19] Denis Mayr Lima Martins. "Reverse Engineering Database Queries from Examples: State-of-the-art, Challenges, and Research Opportunities." In: *Information Systems* 83 (2019), pages 89–100. DOI: 10.1016/J.IS.2019.03.002 (cited on page 13).

[NT00] Jaroslav Nesetril and Claude Tardif. "Duality Theorems for Finite Structures (Characterising Gaps and Good Characterisations)." In: *Journal of Combinatorial Theory, Series B* 80.1 (2000), pages 80–97. DOI: 10.1006/jctb.2000.1970 (cited on page 194).

[NT05] Jaroslav Nesetril and Claude Tardif. "Short Answers to Exponentially Long Questions: Extremal Aspects of Homomorphism Duality." In: *SIAM Journal on Discrete Mathematics* 19.4 (2005), pages 914–920. DOI: 10.1137/S0895480104445630 (cited on page 183).

[ND97] Shan-Hwei Nienhuys-Cheng and Ronald De Wolf. *Foundations of Inductive Logic Programming*. Volume 1228. Springer Science & Business Media, 1997. URL: https://link.springer.com/book/10.1007/3-540-62927-0 (visited on 05/14/2024) (cited on page 14).

[OEI24] OEIS Foundation Inc. *Entry A000081*. The On-Line Encyclopedia of Integer Sequences. 2024. URL: https://oeis.org/A000081 (visited on 04/16/2024) (cited on pages 180, 202).

[Ort19] Magdalena Ortiz. "Ontology-Mediated Queries from Examples: A Glimpse at the DL-Lite Case." In: *Proceedings of the 5th Global Conference on Artificial Intelligence*. GCAI 2019. Edited by Diego Calvanese and Luca Iocchi. Volume 65. EPiC Series in Computing. EasyChair, 2019, pages 1–14. DOI: 10.29007/jhtz (cited on pages 14, 35).

[Ott06] Martin Otto. "Bisimulation Invariance and Finite Models." In: *Logic Colloquium '02*. Edited by Zoé Chatzidakis, Peter Koepke, and Wolfram Pohlers. Lecture Notes in Logic. 2006, pages 276–298. URL: https://doi.org/10.1017/9781316755723.013 (cited on page 166).

[Oza20] Ana Ozaki. "Learning Description Logic Ontologies Five Approaches: Where Do They Stand." In: *Künstliche Intelligenz* 34.3 (2020), pages 317–327. DOI: 10.1007/s13218-020-00656-9 (cited on page 19).

[OPM20] Ana Ozaki, Cosimo Persia, and Andrea Mazzullo. "Learning Query Inseparable ELH Ontologies." In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI 2020. AAAI Press, 2020, pages 2959–2966. DOI: 10.1609/aaai.v34i03.5688 (cited on pages 18, 43).

[PV88] Leonard Pitt and Leslie G. Valiant. "Computational Limitations on Learning from Examples." In: *Journal of the ACM* 35.4 (1988), pages 965–984. DOI: 10.1145/48014.63140 (cited on pages 41, 170).

[PW90] Leonard Pitt and Manfred K. Warmuth. "Prediction-Preserving Reducibility." In: *Journal of Computer and System Sciences* 41.3 (1990), pages 430–467. DOI: 10.1016/0022-0000(90)90028-J (cited on page 59).

[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 X*. Lecture Notes in Computer Science 4900 (2008), pages 133–173. DOI: 10.1007/978-3-540-77688-8_5 (cited on pages 2, 4, 23).

[Pul22] Hadrien Pulcini. "Logical Separability of Open-World Data." PhD thesis. University of Liverpool, 2022. DOI: 10.17638/03165443 (cited on page 14).

[RH97] Alan Rector and Ian Horrocks. "Experience Building a Large, Re-Usable Medical Ontology Using a Description Logic with Transitivity and Concept Inclusions." In: *Proceedings of the Workshop on Ontological Engineering*. AAAI'97. AAAI Press, 1997, pages 321–325. URL: https://aaai.org/papers/0013-ss97-06-013 (visited on 05/14/2024) (cited on page 4).

[RFdA20] Giuseppe Rizzo, Nicola Fanizzi, and Claudia d'Amato. "Class Expression Induction as Concept Space Exploration: From DL-Foil to DL-Focl." In: *Future Generation Computer Systems* 108 (2020), pages 256–272. ISSN: 0167-739X. DOI: 10.1016/j.future.2020.02.071 (cited on page 15).

[Ros07] Riccardo Rosati. "On Conjunctive Query Answering in EL." In: *Proceedings of the 2007 International Workshop on Description Logics*. DL 2007. Edited by Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris. Volume 250. CEUR Workshop Proceedings. CEUR-WS.org, 2007. URL: http://ceur-ws.org/Vol-250/paper_83.pdf (visited on 05/14/2024) (cited on page 30).

[SH19] Md Kamruzzaman Sarker and Pascal Hitzler. "Efficient Concept Induction for Description Logics." In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI 2019. AAAI Press, 2019, pages 3036–3043. DOI: 10.1609/aaai.v33i01.33013036 (cited on page 15).

[Sch+09] Stefan Schulz, Boontawee Suntisrivaraporn, Franz Baader, and Martin Boeker. "SNOMED Reaching Its Adolescence: Ontologists' and Logicians' Health Check." In: *International Journal of Medical Informatics* 78 (Supplement-1 2009), S86–S94. DOI: 10.1016/J.IJMEDINF.2008.06.004 (cited on page 4).

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. 1st edition. Cambridge University Press, 2014. ISBN: 978-1-107-29801-9. DOI: 10.1017/CBO9781107298019 (cited on page 176).

[SBM20] Gunjan Singh, Sumit Bhatia, and Raghava Mutharaju. "OWL2Bench: A Benchmark for OWL 2 Reasoners." In: *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference*. ISWC 2020. Edited by Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof

Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal. Volume 12507. Lecture Notes in Computer Science. Springer, 2020, pages 81–96. DOI: 10.1007/978-3-030-62466-8_6 (cited on page 206).

[SCC97] Kent A. Spackman, Keith E. Campbell, and Roger A. Côté. "SNOMED RT: A Reference Terminology for Health Care." In: *American Medical Informatics Association Annual Symposium*. AMIA 1997. AMIA, 1997, pages 640–644. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2233423/pdf/procamiaafs00001-0675.pdf (visited on 05/14/2024) (cited on page 4).

[Suc+23] Fabian M. Suchanek, Mehwish Alam, Thomas Bonald, Pierre-Henri Paris, and Jules Soria. *Integrating the Wikidata Taxonomy into YAGO*. 2023. DOI: 10.48550/ARXIV.2308.11884. arXiv: 2308.11884. Prepublished (cited on page 2).

[TWS20] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. "YAGO 4: A Reason-Able Knowledge Base." In: *The Semantic Web - 17th International Conference*. ESWC 2020. Edited by Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez. Volume 12123. Lecture Notes in Computer Science. Springer, 2020, pages 583–596. DOI: 10.1007/978-3-030-49461-2_34 (cited on page 204).

[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: *Fundamenta Informaticae* 133.2–3 (2014), pages 287–303. DOI: 10.3233/FI-2014-1077 (cited on page 15).

[TZ13] Anni-Yasmin Turhan and Benjamin Zarrieß. "Computing the Lcs w.r.t. General EL+-TBoxes." In: *Informal Proceedings of the 26th International Workshop on Description Logics*. DL 2013. Volume 1014. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pages 477–488. URL: https://ceur-ws.org/Vol-1014/paper_26.pdf (visited on 05/14/2024) (cited on page 103).

[Val84] Leslie G. Valiant. "A Theory of the Learnable." In: *Communications of the ACM* 27.11 (1984), pages 1134–1142. DOI: 10.1145/1968.1972 (cited on pages 7, 40).

[Wes+19] Patrick Westphal, Lorenz Bühmann, Simon Bin, Hajira Jabeen, and Jens Lehmann. "SML-Bench - A Benchmarking Framework for Structured Machine Learning." In: *Semantic Web* 10.2 (2019), pages 231–245. DOI: 10.3233/SW-180308 (cited on pages 199, 204).

[Wil10] Ross Willard. "Testing Expressibility Is Hard." In: *Principles and Practice of Constraint Programming - 16th International Conference*. CP 2010. Edited by David Cohen. Volume 6308. Lecture Notes in Computer Science. Springer, 2010, pages 9–23. DOI: 10.1007/978-3-642-15396-9_4 (cited on page 34).

[Xia+18] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. "Ontology-Based Data Access: A Survey." In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI-ECAI 2018. Edited by Jérôme Lang. International Joint Conferences on Artificial Intelligence, 2018, pages 5511–5519. DOI: 10.24963/IJCAI.2018/777 (cited on pages 2, 4).

[Yan81] Mihalis Yannakakis. "Algorithms for Acyclic Database Schemes." In: *Very Large Data Bases, 7th International Conference*. VLDB '81. IEEE Computer Society, 1981, pages 82–94. URL: https://dl.acm.org/doi/10.5555/1286831.1286840 (visited on 05/14/2024) (cited on pages 30, 173).

[ZT13] Benjamin Zarrieß and Anni-Yasmin Turhan. "Most Specific Generalizations w.r.t. General EL-TBoxes." In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI 2013. Edited by Francesca Rossi. Menlo Park, California: AAAI Press/International Joint Conferences on Artificial Intelligence, 2013, pages 1191–1197. URL: https://www.ijcai.org/Proceedings/13/Papers/179.pdf (visited on 05/14/2024) (cited on page 13).

[Zho+13] Yujiao Zhou, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, and Jay Banerjee. "Making the Most of Your Triple Store: Query Answering in OWL 2 Using an RL Reasoner." In: *22nd International World Wide Web Conference*. WWW '13. Edited by Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo Baeza-Yates, and Sue B. Moon. International World Wide Web Conferences Steering Committee / ACM, 2013, pages 1569–1580. DOI: 10.1145/2488388.2488525 (cited on page 206).

[Zlo75a] Moshé M. Zloof. "Query by Example." In: *American Federation of Information Processing Societies: 1975 National Computer Conference, 19-22 May 1975, Anaheim, CA, USA*. Volume 44. AFIPS Conference Proceedings. AFIPS Press, 1975, pages 431–438. DOI: 10.1145/1499949.1500034 (cited on page 1).

[Zlo75b] Moshé M. Zloof. "Query-by-Example: The Invocation and Definition of Tables and Forms." In: *Proceedings of the International Conference on Very Large Data Bases*. VLDB '75. Edited by Douglas S. Kerr. ACM, 1975, pages 1–24. DOI: 10.1145/1282480.1282482 (cited on page 1).

Index of Notation and Symbols

$\top$	Concept tautology	Page 21
$C \sqcap D$	Conjunction	Page 21
$\exists R.C$	Existential restriction	Page 21
$C \sqsubseteq D$	Concept subsumption	Page 22
$\overline{a}$	Tuple	Page 24
$\mathsf{ind}(\mathcal{A})$	Set of individual names in ABox $\mathcal{A}$	Page 24
$\mathcal{A}, \mathcal{O} \models C(a)$	a is in the extension of the concept C in every model of $\mathcal{A}$ and $\mathcal{O}$	Page 25
$\mathcal{I}_1, \overline{d} \to \mathcal{I}_2, \overline{e}$	Existence of a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$ that maps $\overline{d}$ to $\overline{e}$	Page 26
$\mathsf{var}(q)$	Set of variables of CQ q	Page 25
$\mathsf{img}(h)$	Image of function h	Page 25
$q \subseteq_{\mathcal{O}} p$	Query implication under ontology $\mathcal{O}$	Page 26
$q \equiv_{\mathcal{O}} p$	Query equivalence under ontology $\mathcal{O}$	Page 26
$\mathcal{I} \models q(\overline{d})$	$\overline{d}$ is an answer to CQ q in $\mathcal{I}$	Page 26
$\mathcal{A}, \mathcal{O} \models q(\overline{d})$	$\overline{d}$ is an answer to CQ q in every model of ABox $\mathcal{A}$ and ontology $\mathcal{O}$	Page 26
$\mathsf{sig}(o)$	Signature of o	Page 28
$\|\|o\|\|$	Size of o	Page 28
q_x	Subquery of ELIQ q rooted at x	Page 28
$\mathcal{Q}_\Sigma$	Class of all $\mathcal{Q}$ queries that only use symbols from Σ	Page 28
$\mathcal{I}_1 \times \mathcal{I}_2$	Direct product of $\mathcal{I}_1$ and $\mathcal{I}_2$	Page 28
$\overline{x} \otimes \overline{y}$	Tuple of pairs	Page 28

$a \rightsquigarrow^{R}_{\mathcal{A},\mathcal{O}} M$	$\mathcal{A}$ and $\mathcal{O}$ imply the existence of an R successor of a	Page 31
$\mathsf{error}_{P,q_T,\mathcal{O}}(q)$	Error of q relative to q_T under $\mathcal{O}$	Page 40
q^{-o^x}	CQ q without variable x	Page 62
$\mathcal{U}_{\mathcal{A},\mathcal{O}}$	Universal model of ABox $\mathcal{A}$ and ontology $\mathcal{O}$	Page 31
$\mathcal{I}_1, d_1 \preceq_{\mathcal{ELI}} \mathcal{I}_2, d_2$	Existence of an $\mathcal{ELI}$ simulation from $\mathcal{I}_1$ to $\mathcal{I}_2$ that includes d_1 and d_2	Page 89
$C_{\mathcal{A},\mathcal{O}}$	Compact model of ABox $\mathcal{A}$ and ontology $\mathcal{O}$ that is ELQ-universal	Page 116
$\mathrm{CQ}^{\mathrm{csf}}$	Chordal and symmetry-free CQ	Page 119
$\mathrm{ELIQ}^{\mathrm{sf}}$	Symmetry-free ELIQ	Page 119
$\mathsf{dist}_q(x, y)$	Distance of variables x and y in q	Page 118
$C^3_{\mathcal{A},\mathcal{O}}$	3-compact model of ABox $\mathcal{A}$ and ontology $\mathcal{O}$ that is $\mathrm{CQ}^{\mathrm{csf}}$-universal	Page 120
$\mathcal{I}_1, d_1 \preceq_{\mathcal{EL}} \mathcal{I}_2, d_2$	Existence of an $\mathcal{EL}$ simulation from $\mathcal{I}_1$ to $\mathcal{I}_2$ that includes d_1 and d_2	Page 121
$h^*(x)$	Function that maps x to the root of the trace $h(x)$	Page 126

List of Figures

List of Tables

Bibliographische Daten

Titel	Learning Queries under Description Logic Ontologies
Typ	Dissertation
Autor	Maurice Funk
Jahr	2024
Sprache	Englisch
Seiten	255
Abbildungen	30
Keywords	Description Logic Ontologies, Conjunctive Queries, Exact Learning, PAC Learning

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 19.06.2024

(Ort, Datum)

(Unterschrift)