7
Query Answering

An important application of ontologies is to provide semantics and domain knowledge for data. Traditionally, data has been stored and managed inside relational database systems (aka SQL databases) where it is organised according to a pre-specified schema that describes its structure and meaning. In recent years, though, less and less data comes from such controlled sources. In fact, a lot of data is now found on the web, in social networks and so on, where typically neither its structure nor its meaning is explicitly specified; moreover, data coming from such sources is typically highly incomplete. Ontologies can help to overcome these problems by providing semantics and background knowledge, leading to a paradigm that is often called *ontology-mediated querying*. As an example, consider data about used-car offers. The ontology can add knowledge about the domain of cars, stating for example that a grand tourer is a kind of sports car. In this way, it becomes possible to return a car that the data identifies as a grand tourer as an answer to a query which asks for finding all sports cars. In the presence of data, a fundamental description logic reasoning service is answering database queries in the presence of ontologies. Since answers to full SQL queries are uncomputable in the presence of ontologies, the prevailing query language is conjunctive queries (CQs) and slight extensions thereof such as unions of conjunctive queries (UCQs) and positive existential queries. Conjunctive queries are essentially the select-from-where fragment of SQL, written in logic.

In this chapter, we study conjunctive query answering in the presence of ontologies that take the form of a DL TBox. In particular, we show how to implement this reasoning service using standard database systems such as relational (SQL) systems and Datalog engines, taking advantage of those systems' efficiency and maturity. Since database sys-

tems are not prepared to deal with TBoxes, we need a way to "sneak them in". While there are several approaches to achieve this, here we will concentrate on *query rewriting*: given a CQ q to be answered and a TBox $\mathcal{T}$, produce a query $q_{\mathcal{T}}$ such that, for any ABox $\mathcal{A}$, the answers to q on $\mathcal{A}$ and $\mathcal{T}$ are identical to the answers to $q_{\mathcal{T}}$ given by a database system that stores $\mathcal{A}$ as data. Thus, query rewriting can be thought of as integrating the TBox into the query. Different query languages for $q_{\mathcal{T}}$ such as SQL and Datalog give rise to different query rewriting problems. In general, it turns out that rewritten queries are guaranteed to exist only when the TBox is formulated in a very inexpressive DL. When rewriting into SQL queries, rewritings are in fact not guaranteed to exist for any of the DLs discussed in the earlier chapters of this book. This observation leads to the introduction of the DL-Lite family of description logics that was designed specifically to guarantee the existence of SQL rewritings. Rewriting into Datalog instead of into SQL enables the use of more expressive DLs for formulating the TBox. In fact, rewritings are guaranteed to exist when the TBox is formulated in $\mathcal{EL}$, $\mathcal{ELI}$ and several extensions thereof.

7.1 Conjunctive queries and FO queries

We introduce and discuss the essentials of conjunctive queries, starting with their syntax.

Definition 7.1 (Conjunctive query)**.** Let $\mathbf{V}$ be a set of *variables*. A *term* t is a variable from $\mathbf{V}$ or an individual name from $\mathbf{I}$.

A *conjunctive query* (CQ) q has the form $\exists x_1 \cdots \exists x_k \, (\alpha_1 \wedge \cdots \wedge \alpha_n)$, where $k \geq 0$, $n \geq 1$, $x_1, \ldots, x_k \in \mathbf{V}$, and each α_i is a *concept atom* $A(t)$ or a *role atom* $r(t, t')$ with $A \in \mathbf{C}$, $r \in \mathbf{R}$, and t, t' *terms*.

We call $x_1, \ldots, x_k$ *quantified variables* and all other variables in q, *answer variables*. The *arity* of q is the number of answer variables.

To express that the answer variables in a CQ q are $\vec{x}$, we often write $q(\vec{x})$ instead of just q. Here are a number of simple examples of conjunctive queries; for easy identification, answer variables are underlined.

(i) Return all pairs of individual names (a, b) such that a is a professor who supervises student b:

$$q_1(x_1, x_2) = \mathsf{Professor}(\underline{x_1}) \wedge \mathsf{supervises}(\underline{x_1}, \underline{x_2}) \wedge \mathsf{Student}(\underline{x_2}).$$

(ii) Return all individual names a such that a is a student supervised by some professor:

$$q_2(x) = \exists y\, (\mathsf{Professor}(y) \wedge \mathsf{supervises}(y, \underline{x}) \wedge \mathsf{Student}(\underline{x})).$$

(iii) Return all pairs of students supervised by the same professor:

$$q_3(x_1, x_2) = \exists y\, (\mathsf{Professor}(y) \wedge \mathsf{supervises}(y, \underline{x_1}) \wedge \mathsf{supervises}(y, \underline{x_2}) \wedge \mathsf{Student}(\underline{x_1}) \wedge \mathsf{Student}(\underline{x_2})).$$

(iv) Return all students supervised by professor smith (an individual name):

$$q_4(x) = (\mathsf{supervises}(\mathsf{smith}, \underline{x}) \wedge \mathsf{Student}(\underline{x})).$$

Observe that every conjunctive query q returns tuples of individual names $(a_1, \ldots, a_k)$, where k is the arity of q. Each such tuple is called an *answer* to the query. To formally define the semantics of CQs, we need to make precise which tuples of individual names qualify as answers. This is done in two steps: first on the level of interpretations and then on the level of knowledge bases.

Definition 7.2. Let q be a conjunctive query and $\mathcal{I}$ an interpretation. We use $\mathsf{term}(q)$ to denote the terms in q. A *match of q in $\mathcal{I}$* is a mapping $\pi : \mathsf{term}(q) \to \Delta^{\mathcal{I}}$ such that

- $\pi(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{term}(q) \cap \mathbf{I}$,
- $\pi(t) \in A^{\mathcal{I}}$ for all concept atoms $A(t)$ in q, and
- $(\pi(t_1), \pi(t_2)) \in r^{\mathcal{I}}$ for all role atoms $r(t_1, t_2)$ in q.

Let $\vec{x} = x_1, \ldots, x_k$ be the answer variables in q and $\vec{a} = a_1, \ldots, a_k$ be individual names from $\mathbf{I}$. We call the match π of q in $\mathcal{I}$ an $\vec{a}$-match if $\pi(x_i) = a_i^{\mathcal{I}}$ for $1 \leq i \leq k$. Then $\vec{a}$ is an *answer to q on $\mathcal{I}$* if there is an $\vec{a}$-match π of q in $\mathcal{I}$. We use $\mathsf{ans}(q, \mathcal{I})$ to denote the set of all answers to q on $\mathcal{I}$.

Consider, for example, the interpretation $\mathcal{I}$ in Figure 7.1, where we assume for simplicity that all individual names are interpreted as themselves, as for example in $\mathsf{mark}^{\mathcal{I}} = \mathsf{mark}$. Then there are three answers to the above query $q_2(x)$ on $\mathcal{I}$, which are mark, alex, and lily. There are seven answers to $q_3(x_1, x_2)$ on $\mathcal{I}$, including (mark, alex), (alex, lily), (lily, alex) and (mark, mark). As illustrated by the last answer, a match need not be injective. Also note that, mathematically, a match is nothing but a *homomorphism* from the query (viewed as a graph) to the interpretation (also viewed as a graph). We now lift the notion of an answer

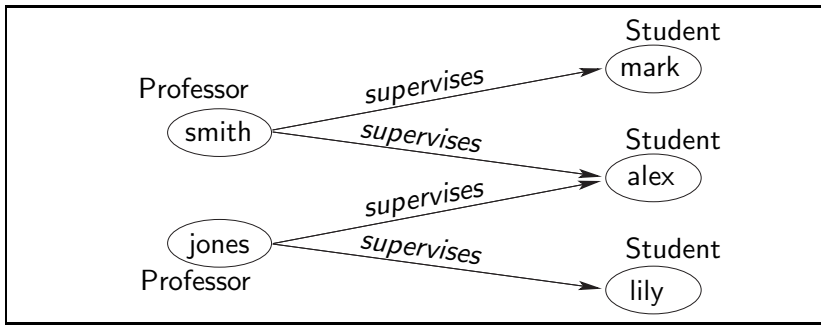

Fig. 7.1. An example interpretation $\mathcal{I}$.

from interpretations to knowledge bases, which have many possible interpretations as models. Thus, when querying a KB we are interested in querying a *set of interpretations* instead of only a single interpretation. In such a situation, so-called *certain answers* provide a natural semantics.

Definition 7.3. Let $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ be a knowledge base. Then $\vec{a}$ is a *certain answer to q on $\mathcal{K}$* if all individual names from $\vec{a}$ occur in $\mathcal{A}$ and $\vec{a} \in \mathsf{ans}(q, \mathcal{I})$ for every model $\mathcal{I}$ of $\mathcal{K}$. We use $\mathsf{cert}(q, \mathcal{K})$ to denote the set of all certain answers to q on $\mathcal{K}$; that is, $\mathsf{cert}(q, \mathcal{K}) = \bigcap_{\mathcal{I} \text{ model of } \mathcal{K}} \mathsf{ans}(q, \mathcal{I})$.

As an example, consider the following knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ formulated in $\mathcal{ALCI}$:

$$\begin{aligned} \mathcal{T} &= \{\mathsf{Student} \sqsubseteq \exists \mathsf{supervises}^{-}.\mathsf{Professor}\}, \\ \mathcal{A} &= \{\mathsf{smith} : \mathsf{Professor}, \mathsf{mark} : \mathsf{Student}, \mathsf{alex} : \mathsf{Student}, \mathsf{lily} : \mathsf{Student}, \\ &\quad (\mathsf{smith}, \mathsf{mark}) : \mathsf{supervises}, (\mathsf{smith}, \mathsf{alex}) : \mathsf{supervises}\}. \end{aligned}$$

Note that the interpretation in Figure 7.1 is a model of this KB. Let us first consider the query $q_4(x)$ from above. As expected, we have $\mathsf{cert}(q_4, \mathcal{K}) = \{\mathsf{mark}, \mathsf{alex}\}$. It is easy to find models of $\mathcal{K}$ in which smith supervises more students than mark and alex, but the latter are the only two students on whose supervision by smith *all* models are in agreement. It is illustrative to consider the role of domain elements whose existence is enforced by existential restrictions in the TBox. For the query $q_2(x)$, we find $\mathsf{cert}(q_2, \mathcal{K}) = \{\mathsf{mark}, \mathsf{alex}, \mathsf{lily}\}$. Note that lily is included because she is a student and thus the TBox enforces that she has a supervisor who is a professor in every model of $\mathcal{K}$. Now consider $q_1(x_1, x_2)$

and note that $\mathsf{cert}(q_1, \mathcal{K}) = \{(\mathsf{smith}, \mathsf{mark}), (\mathsf{smith}, \mathsf{alex})\}$, where lily does not occur. The reason is that different supervisors of lily are possible in different models, and thus there is no answer $(\mathsf{xyz}, \mathsf{lily})$ on which all models agree. In summary, elements that are required to satisfy existential restrictions in the TBox never occur in certain answers, but they can contribute to answers by enabling matches that use them as targets for quantified variables in the query.

We remark that, in Definition 7.3, the condition that all individual names from $\vec{a}$ occur in $\mathcal{A}$ prevents us from sometimes having to return infinitely many uninteresting answers, e.g., when the TBox contains $\top \sqsubseteq A$ and the query is $A(x)$ where otherwise *all* individual names would qualify as an answer.

The main reasoning problem for conjunctive queries is, given a knowledge base $\mathcal{K}$ and a CQ q, to compute the certain answers to q on $\mathcal{K}$. We refer to this reasoning problem as *conjunctive query answering*. To simplify algorithms and proofs, it is often convenient to consider conjunctive queries that do not include any individual names (that is, variables are the only terms that occur), which we call *pure*. As a warm-up exercise, we observe that individual names in queries can always be eliminated.

Lemma 7.4. *Conjunctive query answering can be reduced in polynomial time to answering pure conjunctive queries.*

Proof. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base (formulated in any description logic) and q a conjunctive query that contains individual names. Let Γ be the set of individual names that occur in $\mathcal{A}$ or q, and introduce a fresh concept name A_a and a fresh variable x_a for each $a \in \Gamma$. Extend the ABox $\mathcal{A}$ to a new ABox $\mathcal{A}'$ by adding the assertions $A_a(a)$ for each $a \in \Gamma$. Furthermore, derive q' from q by replacing each $a \in \Gamma$ with x_a in q and adding the concept atom $A_a(x_a)$ for each $a \in \Gamma$. We show that $\mathsf{cert}(q, \mathcal{K}) = \mathsf{cert}(q', \mathcal{K}')$, where $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$.

For the "$\subseteq$" direction, assume that $\vec{a} \notin \mathsf{cert}(q', \mathcal{K}')$. Then there is a model $\mathcal{I}$ of $\mathcal{K}'$ such that $\vec{a}$ is not an answer to q' on $\mathcal{I}$; that is, there is no $\vec{a}$-match π of q' in $\mathcal{I}$. Since $\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{I}$ is also a model of $\mathcal{K}$. Moreover, any match π of q in $\mathcal{I}$ can be extended to a match of q' in $\mathcal{I}$ by setting $\pi(x_a) = a^{\mathcal{I}}$ for all $a \in \Gamma$. Consequently, there is no $\vec{a}$-match π of q in $\mathcal{I}$ (because there is no such match of q'). Thus, $\vec{a}$ is not an answer to q on $\mathcal{I}$, implying $\vec{a} \notin \mathsf{cert}(q, \mathcal{K})$.

For the "$\supseteq$" direction, assume that $\vec{a} \notin \mathsf{cert}(q, \mathcal{K})$. Then, for some model $\mathcal{I}$ of $\mathcal{K}$, there is no $\vec{a}$-match π of q in $\mathcal{I}$. Let $\mathcal{I}'$ be obtained from $\mathcal{I}$ by setting $A_a^{\mathcal{I}'} = \{a^{\mathcal{I}}\}$ for all $a \in \Gamma$. Clearly, $\mathcal{I}'$ is a model of $\mathcal{K}'$.

Moreover, any match π of q' in $\mathcal{I}'$ satisfies $\pi(x_a) = a^{\mathcal{I}}$ since $A_a^{\mathcal{I}'} = \{a^{\mathcal{I}}\}$ and is thus also a match of q in $\mathcal{I}'$ and, as the concept names A_a do not occur in q, a match of q in $\mathcal{I}$. Consequently, there is no $\vec{a}$-match π of q' in $\mathcal{I}'$ and thus $\vec{a} \notin \mathsf{cert}(q', \mathcal{K}')$. □

From now on, we can thus assume, without loss of generality, that conjunctive queries are pure, which we do without further notice. As illustrated by the proof of Lemma 7.4, a tuple $\vec{a}$ *not* being an answer to a query q is always witnessed by a *counter model*, that is, by a model $\mathcal{I}$ of $\mathcal{K}$ such that there is no $\vec{a}$-match π of q in $\mathcal{I}$. In principle, one can thus view conjunctive query answering as a satisfiability problem: a tuple $\vec{a}$ is a certain answer to q on $\mathcal{K}$ if and only if the formula $\mathcal{K}^* \wedge \neg q[\vec{a}/\vec{x}]$ is unsatisfiable, where $\mathcal{K}^*$ is the first-order logic translation of $\mathcal{K}$ described in Section 2.6.1 and $q[\vec{a}/\vec{x}]$ is the first-order sentence obtained from q by consistently replacing the answer variables in $\vec{x}$ with the individual names in $\vec{a}$. Many algorithms for conjunctive query answering in Description Logic are based on this intuition.

We will be interested in query rewriting with SQL as a target language. Because dealing with SQL syntax is too unwieldy for our purposes and since, by Codd's theorem, SQL is equivalent to (a minor restriction of) first-order logic (FO) used as a query language, we instead use the standard syntax of FO.

Definition 7.5 (FO query)**.** An *FO query* is a first-order formula that uses only unary predicates (concept names) and binary predicates (role names), and no function symbols or constants. The use of equality is allowed.

The free variables $\vec{x}$ of an FO query $q(\vec{x})$ are called *answer variables*. The *arity* of $q(\vec{x})$ is the number of answer variables.

Let $q(\vec{x})$ be an FO query of arity k, and $\mathcal{I}$ an interpretation. We say that $\vec{a} = a_1, \ldots, a_k$ is an *answer* to q on $\mathcal{I}$ if $\mathcal{I} \models q[\vec{a}]$; that is, $q(\vec{x})$ evaluates to true in $\mathcal{I}$ under the valuation that interprets the answer variables $\vec{x}$ as $\vec{a}$. We write $\mathsf{ans}(q, \mathcal{T})$ to denote the set of all answers to q in $\mathcal{I}$.

An example FO query of arity one is $A(x) \vee \forall y\,(r(x, y) \rightarrow s(y, x))$. Note that conjunctive queries are a special case of FO queries and that, for every conjunctive query q, the set $\mathsf{ans}(q, \mathcal{I})$ defined in Definition 7.2 agrees with the set set $\mathsf{ans}(q, \mathcal{I})$ defined in Definition 7.5. Computing the answers to an FO query q on an interpretation $\mathcal{I}$ as in Definition 7.5 is exactly the querying service offered by a relational database system,

with $\mathcal{I}$ corresponding to the data stored in the database and q to an SQL query. We use FO queries only as a target language for rewriting, but not to query, knowledge bases because they are too expressive for the latter purpose: it is not hard to reduce satisfiability of FO formulas to answering FO queries on knowledge bases (with an empty TBox), and thus answers to FO queries on knowledge bases are uncomputable.

We will sometimes consider queries without answer variables. Such queries are a bit special in that they do not deliver proper answers but evaluate to true or false.

Definition 7.6 (Boolean queries)**.** A conjunctive query or FO query is called *Boolean* if it has arity zero. For a Boolean FO query q and an interpretation $\mathcal{I}$, we write $\mathcal{I} \models q$ and say that $\mathcal{I}$ *entails* q if the empty tuple is an answer to q on $\mathcal{I}$. For a Boolean conjunctive query q and a knowledge base $\mathcal{K}$, we write $\mathcal{K} \models q$ and say that $\mathcal{K}$ *entails* q if the empty tuple is a certain answer to q on $\mathcal{K}$.

As a simple example, consider the Boolean CQ $\exists x\, \mathsf{Professor}(x)$. For the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ introduced after Definition 7.3, we have $\mathcal{K} \models q$.

7.2 FO-rewritability and DL-Lite

Of course, it is possible to start from scratch when developing algorithms and systems for answering conjunctive queries in the presence of Description Logic knowledge bases, and this has in fact been done for many of the DLs treated in this book. However, conjunctive query answering is most useful in database-style applications where there is a huge amount of data, stored in the ABox. Even without a TBox, efficiently answering queries over large amounts of data is a challenging engineering enterprise, and the TBox makes it all the more difficult. It is thus a natural idea to make use of existing database systems for query answering. The obvious challenge is to accommodate the TBox, which the relational database system is not prepared to process. There are two fundamental ways in which a TBox can be sneaked into a relational database system, as well as various possible variations and combinations thereof. One approach is to replace the original ABox (which is now simply the data stored in the relational system) with a new ABox that includes all the consequences of the TBox, leaving untouched the query to be answered. This approach is often called *materialisation*. The second approach is to leave the data untouched and instead anticipate the consequences of the

TBox in the query. Here, we discuss only the second approach, known as *query rewriting*.

7.2.1 Introducing DL-Lite

In the query rewriting approach to conjunctive query answering, we aim to construct, given a TBox $\mathcal{T}$ and a conjunctive query q, a new query $q_{\mathcal{T}}$ such that, for any ABox $\mathcal{A}$, the certain answers to q on $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ are exactly the answers that a relational database system returns when executing $q_{\mathcal{T}}$ on $\mathcal{A}$ stored as a relational dataset. With the exception of the preprocessing step of constructing $q_{\mathcal{T}}$, we can thus completely delegate query answering to the database system. We take a rather abstract view of relational database systems here, assuming that a relational dataset is simply a DL interpretation and that the queries that the system is able to process are exactly first-order (FO) queries as defined above. Consequently, we call the query $q_{\mathcal{T}}$ an *FO-rewriting* of q with respect to $\mathcal{T}$.

In the following, we restrict ourselves to *simple* ABoxes where, in all concept assertions $a : C$, the concept C must be a concept *name*. Note that it is always possible to make an ABox in a KB simple by replacing each concept assertion $a : C$ with $a : A_C$, where A_C is a fresh concept name, and adding $A_C \sqsubseteq C$ to the TBox. Simple ABoxes can be viewed as an interpretation $\mathcal{I}_{\mathcal{A}}$ and thus stored in a database system by taking $\Delta^{\mathcal{I}_{\mathcal{A}}}$ to be the set of individual names used in $\mathcal{A}$, setting $A^{\mathcal{I}_{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}\}$ for all concept names A, $r^{\mathcal{I}_{\mathcal{A}}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$ for all role names r, and $a^{\mathcal{I}} = a$ for all individual names a. In the remainder of Chapter 7, "*ABox*" always means *simple ABox*.

Definition 7.7. Let $\mathcal{T}$ be a TBox and q a conjunctive query. An FO query $q_{\mathcal{T}}$ is an *FO-rewriting* of q with respect to $\mathcal{T}$ if, for all ABoxes $\mathcal{A}$, we have $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}})$ whenever $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent.

As a first example of an FO-rewriting, consider the following TBox and conjunctive query:

$$\mathcal{T}_1 = \{B_1 \sqsubseteq A, B_2 \sqsubseteq A\}, \qquad q_1(x) = A(x).$$

It is not difficult to see that the FO query $A(x) \vee B_1(x) \vee B_2(x)$ is an FO-rewriting of q_1 with respect to $\mathcal{T}_1$. In fact, if the ABox contains the assertion $a : A$, then trivially $a \in \mathsf{cert}(q_1, \mathcal{K})$, which explains the first disjunct $A(x)$ in the rewriting. However, $a : B_1$ in the ABox also yields $a \in \mathsf{cert}(q_1, \mathcal{K})$ because of the inclusion $B_1 \sqsubseteq A$ in $\mathcal{T}_1$, which explains

the second disjuct $B_1(x)$, and likewise for $B_2(x)$. As a second example, consider

$$\mathcal{T}_2 = \{A \sqsubseteq \exists r.A\}, \qquad q_2(x) = A(x).$$

In this case, q_2 itself is an FO-rewriting of q_2 with respect to $\mathcal{T}_2$, that is, we can simply ignore the TBox. The reason is that, although $\mathcal{T}_2$ might imply the existence of additional individuals that are instances of A, we have already seen that elements required to satisfy existential restrictions can never be returned as an answer. It is interesting to contrast this example with the rather similar TBox and query used in the following result which, slightly disturbingly, shows that even for very simple TBoxes and conjunctive queries, FO-rewritings are not guaranteed to exist.

Theorem 7.8. *There is no FO-rewriting of the conjunctive query* $q(x) = A(x)$ *with respect to the* $\mathcal{EL}$ *TBox* $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$.

Proof. We only provide a sketch. It is not difficult to see that an FO-rewriting $\varphi(x)$ of q with respect to $\mathcal{T}$ has to satisfy $\mathcal{I} \models \varphi[d]$ exactly for those elements d of an interpretation $\mathcal{I}$ that reach an A-element along an r-chain, that is, there are elements $d_0, \dots, d_n$ such that $d = d_0$, $(d_i, d_{i+1}) \in r^{\mathcal{I}}$ for all $i < n$, and $d_n \in A^{\mathcal{I}}$. It is well known that reachability properties of this sort are not expressible in first-order logic.

More specifically, we can use Gaifman locality to prove that there is no FO query $\varphi(x)$ with the above property. Let $\mathcal{I}$ be an interpretation and $d \in \Delta^{\mathcal{I}}$. For $k \geq 0$, the *k-neighbourhood* around d in $\mathcal{I}$, denoted $N^k_{\mathcal{I}}(d)$, is defined as the restriction of $\mathcal{I}$ to those elements that are reachable from d along a role chain of length at most k. It follows from a classical result of Gaifman that, for every FO query $\varphi(x)$, there is a number k such that the following holds: for all interpretations $\mathcal{I}$ and $d_1, d_2 \in \Delta^{\mathcal{I}}$ with $N^k_{\mathcal{I}}(d_1) = N^k_{\mathcal{I}}(d_2)$, we have $\mathcal{I} \models \varphi[d_1]$ if and only if $\mathcal{I} \models \varphi[d_2]$. Now assume that the desired FO-rewriting $\varphi(x)$ of q with respect to $\mathcal{T}$ exists and let k be the mentioned number. Take an interpretation $\mathcal{I}$ that is the disjoint union of two r-chains of length $k+1$. The first one begins at element d_1 and ends at e_1 while the second one begins at d_2 and ends at e_2. Assume that $A^{\mathcal{I}} = \{e_1\}$ and thus in particular $e_2 \notin A^{\mathcal{I}}$. By the desired property of $\varphi(x)$, we should have $\mathcal{I} \models \varphi[d_1]$ and $\mathcal{I} \not\models \varphi[d_1]$. However, this contradicts Gaifman's observation and the fact that $N^k_{\mathcal{I}}(d_1) = N^k_{\mathcal{I}}(d_2)$. □

Theorem 7.8 casts serious doubt on the feasibility of the query rewriting approach to conjunctive query answering: FO-rewritings are not

guaranteed to exist even when we confine ourselves to the tractable and moderately expressive description logic $\mathcal{EL}$. In fact, the proof of Theorem 7.8 illustrates that it is the recursive nature of the concept inclusion $\exists r.A \sqsubseteq A$ that conflicts with Gaifman locality and thus also with FO-rewritability. To make the query rewriting approach work, we therefore have to use a description logic that avoids such forms of recursion. The DL-Lite family of DLs has been introduced specifically for this purpose. In the following, we introduce one typical member of this family.

Definition 7.9. All of the following are *basic DL-Lite concepts*:

- every concept name,
- $\top$ (the top concept),
- $\exists r$ (unqualified existential restriction), and
- $\exists r^-$ (unqualified existential restriction on inverse role).

A *DL-Lite TBox* is a finite set of

- positive concept inclusions $B_1 \sqsubseteq B_2$,
- negative concept inclusions $B_1 \sqsubseteq \neg B_2$,
- role inclusion axioms $r \sqsubseteq s$,

where B_1 and B_2 range over basic DL-Lite concepts and r and s over role names and their inverses.

The above version of DL-Lite is a slight restriction of what in the literature is known as DL-Lite$_\mathcal{R}$.[1] We drop the subscript, which indicates the presence of role inclusions, for readability. The DL-Lite concept $\exists r$ is an abbreviation for $\exists r.\top$, which also clarifies its semantics (likewise for $\exists r^-$). Thus, DL-Lite replaces full existential restrictions with an *unqualified* version; that is, we can speak about the existence of an r-successor, but cannot further qualify its properties. Note that DL-Lite does not allow unbounded syntactic nesting of concept expressions.

The following is an example of a DL-Lite TBox; it consists of six concept inclusions and one role inclusion:

$$\begin{aligned}
\mathsf{Professor} &\sqsubseteq \mathsf{Teacher}, & \mathsf{Teacher} &\sqsubseteq \mathsf{Person},\\
\mathsf{Teacher} &\sqsubseteq \exists \mathsf{teaches}, & \mathsf{Course} &\sqsubseteq \neg \mathsf{Person},\\
\exists \mathsf{teachesCourse}^- &\sqsubseteq \mathsf{Course}, & \mathsf{Course} &\sqsubseteq \exists \mathsf{teachesCourse}^-,\\
\mathsf{teachesCourse} &\sqsubseteq \mathsf{teaches}.
\end{aligned}$$

[1] The restriction is that, for simplicity, we do not include negative role inclusions.

Suppose we want to answer the following CQ, which asks to return all persons that teach a course:

$$q(x) = \exists y\, \mathsf{Person}(x) \land \mathsf{teaches}(x, y) \land \mathsf{Course}(y).$$

The following is an FO-rewriting of $q(x)$ with respect to $\mathcal{T}$:

$$(\mathsf{Teacher}(x) \lor \mathsf{Professor}(x) \lor \mathsf{Person}(x)) \\ \land ((\mathsf{teaches}(x, y) \land \mathsf{Course}(y)) \lor \mathsf{teachesCourse}(x, y)).$$

Although DL-Lite is a seriously restricted language, a TBox such as the one above can still describe important aspects of the application domain. In fact, DL-Lite can capture the most important aspects of prominent conceptual modelling formalisms such as entity–relationship (ER) diagrams and UML class diagrams.

The most important property of DL-Lite is that FO-rewritings of conjunctive queries with respect to DL-Lite TBoxes are always guaranteed to exist and can often be constructed efficiently. Before we dig into this, we first take a brief look at the more basic reasoning problems of satisfiability and subsumption, which do not involve ABox data. In DL-Lite, satisfiability and subsumption (of basic concepts) turn out to be simple problems, both conceptually and computationally. As in more expressive DLs, subsumption and unsatisfiability are mutually reducible in polynomial time: deciding whether a subsumption $\mathcal{T} \models B_1 \sqsubseteq B_2$ holds is equivalent to deciding whether A is unsatisfiable with respect to $\mathcal{T} \cup \{A \sqsubseteq B_1, A \sqsubseteq \neg B_2\}$, where A is a fresh concept name. Conversely, a basic concept B is unsatisfiable with respect to $\mathcal{T}$ if and only if $\mathcal{T} \models B \sqsubseteq A$, where A is again a fresh concept name. We can thus concentrate on deciding satisfiability, for which the following closure operation is fundamental (NI stands for "negative inclusions"). As in Chapter 4, we use $\mathsf{Inv}(r)$ to denote r^- if r is a role name and s if $r = s^-$.

Definition 7.10. Let $\mathcal{T}$ be a DL-Lite TBox. The *NI-closure* of $\mathcal{T}$, denoted $\mathcal{T}^{\mathsf{NI}}$, is the TBox obtained by starting with $\mathcal{T}$ and then exhaustively applying the following rules:

- C1 If $\mathcal{T}^{\mathsf{NI}}$ contains $\top \sqsubseteq \neg B$, then add $B \sqsubseteq \neg B$;
- C2 If $\mathcal{T}^{\mathsf{NI}}$ contains $\top \sqsubseteq \neg\top$ and B is a basic concept that occurs in $\mathcal{T}$, then add $B \sqsubseteq \neg B$;
- C3 If $\mathcal{T}^{\mathsf{NI}}$ contains $B_1 \sqsubseteq \neg B_2$, then add $B_2 \sqsubseteq \neg B_1$;
- C4 if $\mathcal{T}^{\mathsf{NI}}$ contains $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq \neg B_3$, then add $B_1 \sqsubseteq \neg B_3$;
- C5 if $\mathcal{T}^{\mathsf{NI}}$ contains $B_1 \sqsubseteq B_2$ and $B_2 \sqsubseteq \neg B_2$, then add $B_1 \sqsubseteq \neg B_1$;

C6 if $\mathcal{T}^{\mathsf{NI}}$ contains $B \sqsubseteq \exists r$ and $\exists \mathsf{Inv}(r) \sqsubseteq \neg\exists \mathsf{Inv}(r)$, then add $B \sqsubseteq \neg B$;

C7 if $\mathcal{T}^{\mathsf{NI}}$ contains $r \sqsubseteq s$ and $\exists s \sqsubseteq \neg B$, then add $\exists r \sqsubseteq \neg B$;

C8 if $\mathcal{T}^{\mathsf{NI}}$ contains $r \sqsubseteq s$ and $\exists \mathsf{Inv}(s) \sqsubseteq \neg B$, then add $\exists \mathsf{Inv}(r) \sqsubseteq \neg B$;

C9 if $\mathcal{T}^{\mathsf{NI}}$ contains $r \sqsubseteq s$ and $\exists s \sqsubseteq \neg\exists s$ or $\exists \mathsf{Inv}(s) \sqsubseteq \neg\exists \mathsf{Inv}(s)$, then add $\exists r \sqsubseteq \neg\exists r$.

It is sufficient to decide the satisfiability of concept names instead of basic concepts because a basic concept B is satisfiable with respect to a TBox $\mathcal{T}$ if and only if A_B is satisfiable with respect to $\mathcal{T} \cup \{A_B \sqsubseteq B\}$, where A_B is a fresh concept name. The following result shows that deciding the satisfiability of concept names with respect to a DL-Lite TBox $\mathcal{T}$ merely requires a lookup in $\mathcal{T}^{\mathsf{NI}}$.

Theorem 7.11. *Let $\mathcal{T}$ be a DL-Lite TBox and A_0 a concept name. Then A_0 is satisfiable with respect to $\mathcal{T}$ if and only if $A_0 \sqsubseteq \neg A_0 \notin \mathcal{T}^{\mathsf{NI}}$.*

For proving the (contrapositive of the) "only if" direction, it is enough to prove that the rules applied in the construction of $\mathcal{T}^{\mathsf{NI}}$ are sound – that is, if such a rule is applied adding an inclusion $B_1 \sqsubseteq \neg B_2$ – then $\mathcal{T} \models B_1 \sqsubseteq \neg B_2$. This is straightforward using induction on the number of rule applications, a case distinction according to the rule applied, and the semantics of DL-Lite. The "if" direction is less straightforward and we defer its proof to Section 7.2.2. It is easy to see that, since the rules do not introduce any new basic concepts, the size of $\mathcal{T}^{\mathsf{NI}}$ is at most quadratic in the size of $\mathcal{T}$. The computation of $\mathcal{T}^{\mathsf{NI}}$ thus only takes polynomial time.

Theorem 7.12. *In DL-Lite, satisfiability and subsumption can be decided in polynomial time.*

It is interesting to note that, since in DL-Lite it is not possible to syntactically nest concepts, the set of all basic concepts that can be formed over a fixed finite signature (a set of concept and role names) is finite, and so is the set of concept and role inclusions. As a consequence, it is possible to effectively make explicit *all* inclusions implied by a DL-Lite TBox $\mathcal{T}$ which are formulated in the signature of $\mathcal{T}$ by finitely extending the TBox. This can be done, for example, by testing subsumption between all basic concepts using Theorem 7.12. The size of the completed TBox will be at most quadratic in the size of the signature, since every DL-Lite inclusion contains at most two concept or role names.

7.2.2 Universal models

We now introduce universal models of DL-Lite knowledge bases, which are a central tool for studying conjunctive query answering in DL-Lite. They will also be useful for proving the "if" direction of Theorem 7.11. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite knowledge base. To construct the universal model $\mathcal{I}_\mathcal{K}$ of $\mathcal{K}$, we start by defining an interpretation $\mathcal{I}_0$ as follows:

$$\begin{aligned} \Delta^{\mathcal{I}_0} &= \mathsf{Ind}(\mathcal{A}), \\ A^{\mathcal{I}_0} &= \{a \in \mathsf{Ind}(\mathcal{A}) \mid A(a) \in \mathcal{A}\}, \\ r^{\mathcal{I}_0} &= \{(a,b) \mid r(a,b) \in \mathcal{A}\}, \\ a^{\mathcal{I}_0} &= a, \end{aligned}$$

where $\mathsf{Ind}(\mathcal{A})$ denotes the set of individual names in $\mathcal{A}$. Next, we apply the concept and role inclusions in $\mathcal{T}$ as rules, obtaining the desired universal model in the limit of the resulting sequence of interpretations $\mathcal{I}_0, \mathcal{I}_1, \ldots$ This sequence is defined by starting with $\mathcal{I}_0$ and then exhaustively applying the following rules:

- R1 if $d \in B^{\mathcal{I}_i}$, $B \sqsubseteq A \in \mathcal{T}$ and $d \notin B^{\mathcal{I}_i}$, then add d to $A^{\mathcal{I}_{i+1}}$;
- R2 if $d \in B^{\mathcal{I}_i}$, $B \sqsubseteq \exists r \in \mathcal{T}$ and $d \notin (\exists r)^{\mathcal{I}_i}$, then add a fresh element f to $\Delta^{\mathcal{I}_{i+1}}$ and (d,f) to $r^{\mathcal{I}_{i+1}}$;
- R3 if $(d,e) \in r^{\mathcal{I}_i}$, $r \sqsubseteq s \in \mathcal{T}$ and $(d,e) \notin s^{\mathcal{I}_i}$, then add (d,e) to $s^{\mathcal{I}_{i+1}}$.

In R2, r can be a role name or inverse thereof, and in the latter case "add (d,f) to $r^{\mathcal{I}_{i+1}}$" means adding (f,d) to $s^{\mathcal{I}_{i+1}}$ if $r = \mathsf{Inv}(s)$. The same is true for R3 and s. If no further rule application is possible after the construction of some interpretation $\mathcal{I}_i$, we simply set $\mathcal{I}_{i+\ell} = \mathcal{I}_i$ for all $\ell > 0$.

Note that applications of R3 might cause applications, previously possible, of R2 to become impossible. By applying the rules in a different order, we can thus obtain different universal models in the limit. To prevent this, we assume that applications of R3 are preferred to applications of R2. To make sure that all possible rule applications are eventually carried out, we assume fairness of application; that is, any rule that is applicable will eventually be applied. It can be proved that, with these assumptions, the interpretation obtained in the limit is unique. In the area of databases, the procedure we have just sketched is known as *the (restricted) chase*.

The *universal model* of $\mathcal{K}$ is the interpretation $\mathcal{I}_\mathcal{K}$ obtained as the limit of the sequence $\mathcal{I}_0, \mathcal{I}_1, \ldots$; that is, $\Delta^{\mathcal{I}_\mathcal{K}} = \bigcup_{i \geq 0} \Delta^{\mathcal{I}_i}$, $A^{\mathcal{I}_\mathcal{K}} = \bigcup_{i \geq 0} A^{\mathcal{I}_i}$ for all concept names A, $r^{\mathcal{I}_\mathcal{K}} = \bigcup_{i \geq 0} r^{\mathcal{I}_i}$ for all role names r, and $a^{\mathcal{I}_\mathcal{K}} = a$

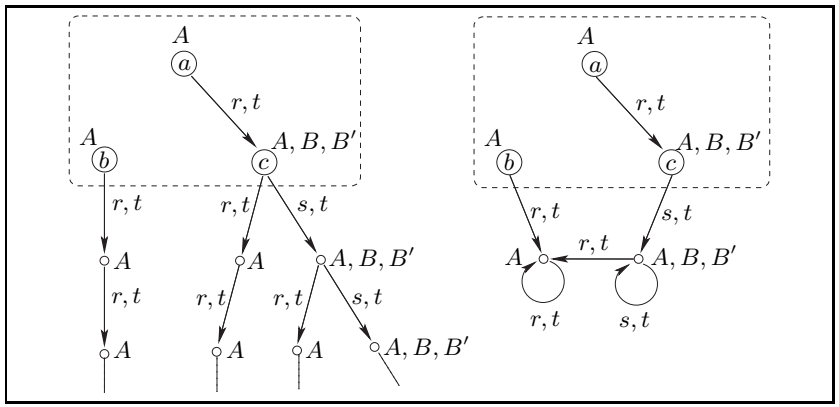

Fig. 7.2. The universal model (left) and another model (right).

for all individual names a. Note that $\mathcal{I}_\mathcal{K}$ might be finite or infinite. As an example, consider the following DL-Lite KB $\mathcal{K}$:

$$A \sqsubseteq \exists r, \qquad B \sqsubseteq \exists s, \qquad \exists t^- \sqsubseteq A, \qquad \exists s^- \sqsubseteq B, \qquad B \sqsubseteq B',$$
$$r \sqsubseteq t, \qquad s \sqsubseteq t,$$
$$a{:}A, \qquad b{:}A, \qquad c{:}B, \qquad (a, c){:}r.$$

The universal model $\mathcal{I}_\mathcal{K}$ is displayed on the left in Figure 7.2 with the dashed box enclosing the domain elements that consistute the ABox part of $\mathcal{K}$; all other domain elements are generated by rule $\mathsf{R2}$ to satisfy existential restrictions. In this case, $\mathcal{I}_\mathcal{K}$ is infinite. On the right-hand side, we show another (finite) model of the same KB $\mathcal{K}$.

We will show shortly that $\mathcal{I}_\mathcal{K}$ is indeed a model of $\mathcal{K}$. However, it is not just *some* model but enjoys special properties which, together with its use for conjunctive query answering later on, earns it the name "universal". The most notable property of $\mathcal{I}_\mathcal{K}$ is that it can be found inside *any* model of $\mathcal{I}_\mathcal{K}$ in terms of a homomorphism; intuitively, this states a form of minimality in the sense that $\mathcal{I}_\mathcal{K}$ makes true only things that need to be true in any model of $\mathcal{K}$.

Definition 7.13. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations. A function $h : \Delta^{\mathcal{I}_1} \to \Delta^{\mathcal{I}_2}$ is a *homomorphism from* $\mathcal{I}_1$ *to* $\mathcal{I}_2$ if the following conditions are satisfied:

(i) $d \in A^{\mathcal{I}_1}$ implies $h(d) \in A^{\mathcal{I}_2}$ for all concept names A;
(ii) $(d, e) \in r^{\mathcal{I}_1}$ implies $(h(d), h(e)) \in r^{\mathcal{I}_2}$ for all role names r; and
(iii) $h(a^{\mathcal{I}_1}) = a^{\mathcal{I}_2}$ for all individual names a.

If there is a homomorphism from $\mathcal{I}_1$ to $\mathcal{I}_2$, we write $\mathcal{I}_1 \to \mathcal{I}_2$.

As an example, consider again Figure 7.2, where it is not hard to find a homomorphism h from the universal model $\mathcal{I}_\mathcal{K}$ on the left-hand side to the model $\mathcal{I}$ on the right-hand side: map all ABox elements to themselves and, outside the ABox, map all elements in $\mathcal{I}_\mathcal{K}$ that satisfy A to the non-ABox element in $\mathcal{I}$ that satisfies A, and similarly for elements satisfying A, B, B'.

We now show that $\mathcal{I}_\mathcal{K}$ indeed behaves as described.

Lemma 7.14. *For every model $\mathcal{I}$ of $\mathcal{K}$, we have $\mathcal{I}_\mathcal{K} \to \mathcal{I}$.*

Proof. Let $\mathcal{I}_0, \mathcal{I}_1, \dots$ be the interpretations used in the construction of $\mathcal{I}_\mathcal{K}$. We show by induction on i that there are $h_0, h_1, \dots$ such that h_i is a homomorphism from $\mathcal{I}_i$ to $\mathcal{I}$ and h_i and h_{i+1} agree on $\Delta^{\mathcal{I}_i}$; that is, $h_{i+1}(d) = h_i(d)$ for all $d \in \Delta^{\mathcal{I}_i}$ (note that $\Delta^{\mathcal{I}_0} \subseteq \Delta^{\mathcal{I}_1} \subseteq \cdots$). The desired homomorphism h from $\mathcal{I}_\mathcal{K}$ to $\mathcal{I}$ is then obtained in the limit as $h = \bigcup_{i \geq 0} h_i$.

For the induction start, the homomorphism h_0 is defined by setting $h_0(a) = a^\mathcal{I}$ for all $a \in \mathsf{Ind}(\mathcal{A})$. Using the fact that $\mathcal{I}$ is a model of $\mathcal{A}$, it is easy to check that conditions (i)–(iii) in Definition 7.13 are satisfied.

For the induction step, assume that h_i has already been defined. To define h_{i+1}, we make a case distinction according to the rule that was applied to obtain $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$:

R1. Then there is a $d \in B^{\mathcal{I}_i}$ and a GCI $B \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding d to the extension of A. We must have $h_i(d) \in B^\mathcal{I}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$, this yields $h_{i+1}(d) \in A^\mathcal{I}$. Consequently, h_i is also a homomorphism from $\mathcal{I}_{i+1}$ to $\mathcal{I}$ and we can set $h_{i+1} = h_i$.

R2. Then there is a $d \in B^{\mathcal{I}_i}$ and a GCI $B \sqsubseteq \exists r \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding a fresh element f to $\Delta^{\mathcal{I}_{i+1}}$ and (d, f) to the extension of r. We must have $h(d) \in B^\mathcal{I}$ and thus $h(d) \in (\exists r)^\mathcal{I}$. By the semantics, we thus find some $(d, e) \in r^\mathcal{I}$. Clearly, $h_{i+1} = h_i \cup \{f \mapsto e\}$ is a homomorphism from $\mathcal{I}_{i+1}$ to $\mathcal{I}$.

R3. Then there is a $(d, e) \in r^{\mathcal{I}_i}$ and a role inclusion $r \sqsubseteq s \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding (d, e) to the extension of s. We must have $(h_i(d), h_i(e)) \in r^\mathcal{I}$ and, since $\mathcal{I}$ is a model of $\mathcal{T}$, also $(h_i(d), h_i(e)) \in s^\mathcal{I}$. Thus we can set $h_{i+1} = h_i$. □

Lemma 7.14 has many interesting applications. In Section 7.2.3, it will play a crucial role in our treatment of conjunctive query answering. It also helps in showing that, as intended, $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$. Of

course, we can only expect this if the knowledge base $\mathcal{K}$ is consistent, as otherwise it does not have any models.

Lemma 7.15. *If $\mathcal{K}$ is consistent, then $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$.*

Proof. Since $\mathcal{A}$ is a simple ABox, the interpretation $\mathcal{I}_0$ from the construction of $\mathcal{I}_\mathcal{K}$ is clearly already a model of $\mathcal{A}$, and therefore so is $\mathcal{I}_\mathcal{K}$. Moreover, all positive concept inclusions and all role inclusions from $\mathcal{T}$ are satisfied in $\mathcal{I}_\mathcal{K}$ since none of the rules $\mathsf{R1}$–$\mathsf{R3}$ is applicable in $\mathcal{I}_\mathcal{K}$. It thus remains to show that, if $\mathcal{K}$ is consistent, then $\mathcal{I}_\mathcal{K}$ also satisfies the negative concept inclusions in $\mathcal{T}$. Assume to the contrary that there is some $B_1 \sqsubseteq \neg B_2 \in \mathcal{T}$ that is not satisfied in $\mathcal{I}_\mathcal{K}$. Then there is a $d \in B_1^{\mathcal{I}_\mathcal{K}} \cap B_2^{\mathcal{I}_\mathcal{K}}$. If $\mathcal{K}$ is consistent, then it has some model $\mathcal{I}$. By Lemma 7.14, there is a homomorphism h from $\mathcal{I}_\mathcal{K}$ to $\mathcal{I}$. But then $h(d) \in B_1^\mathcal{I} \cap B_2^\mathcal{I}$ contradicting that $\mathcal{I}$ is a model of $\mathcal{T}$. □

We close this section by establishing the "only if" direction of Theorem 7.11, which was left open in Section 7.2.1. This completes the proof that satisfiability and subsumption in DL-Lite can be decided in polynomial time. While not using the central Lemma 7.14, the presented proof also relies on universal models.

Lemma 7.16. *Let $\mathcal{T}$ be a DL-Lite TBox and A_0 a concept name. If $A_0 \sqsubseteq \neg A_0 \notin \mathcal{T}^{\mathsf{NI}}$, then A_0 is satisfiable with respect to $\mathcal{T}$.*

Proof. Assume that $A_0 \sqsubseteq \neg A_0 \notin \mathcal{T}^{\mathsf{NI}}$ and consider the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{A} = \{a : A_0\}$ and the universal model $\mathcal{I}_\mathcal{K}$ of $\mathcal{K}$. By construction of the latter, we have $a \in A_0^{\mathcal{I}_\mathcal{K}}$. It thus suffices to show that $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{T}$. Note that we cannot invoke Lemma 7.15 because we do not know whether $\mathcal{K}$ is consistent – in fact this is equivalent to A_0 being satisfiable with respect to $\mathcal{T}$. However, we can argue exactly as in the proof of Lemma 7.15 that $\mathcal{I}_\mathcal{K}$ satisfies all positive concept inclusions and role inclusions in $\mathcal{T}$, since this was independent of $\mathcal{K}$ being consistent. To show that $\mathcal{I}_\mathcal{K}$ also satisfies all negative concept inclusions in $\mathcal{T}$, we prove by induction on i that $\mathcal{I}_i$ satisfies all negative concept inclusions in $\mathcal{T}^{\mathsf{NI}}$, where $\mathcal{I}_0, \mathcal{I}_1, \ldots$ are the interpretations used to construct $\mathcal{I}_\mathcal{K}$. Recall that $\mathcal{T} \subseteq \mathcal{T}^{\mathsf{NI}}$, so working with $\mathcal{T}^{\mathsf{NI}}$ instead of with $\mathcal{T}$ is sufficient.

For the induction start, consider $\mathcal{I}_0$. By definition and since $\mathcal{A} = \{a : A_0\}$, the only negative concept inclusions that $\mathcal{I}_0$ can potentially violate are $\top \sqsubseteq \neg A_0$, $A_0 \sqsubseteq \neg\top$, $\top \sqsubseteq \neg\top$, and $A_0 \sqsubseteq \neg A_0$. By rules $\mathsf{C1}$–$\mathsf{C3}$ for the construction of $\mathcal{T}^{\mathsf{NI}}$, the presence of any of these concept inclusions in $\mathcal{T}^{\mathsf{NI}}$ implies $A_0 \sqsubseteq \neg A_0 \in \mathcal{T}^{\mathsf{NI}}$, contradicting our initial assumption.

For the induction step, consider $\mathcal{I}_{i+1}$. Our aim is to show that if $\mathcal{I}_{i+1}$ violates a negative inclusion in $\mathcal{T}^{\mathsf{NI}}$, then so does $\mathcal{I}_i$, which yields a contradiction to the induction hypothesis. We make a case distinction according to the rule that was applied to obtain $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$:

R1. Then there is a $d \in B^{\mathcal{I}_i}$ and a $B \sqsubseteq A \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding d to the extension of A. This can result in the violation of negative GCIs from $\mathcal{T}^{\mathsf{NI}}$ that are of the form $A \sqsubseteq \neg B'$ or $B' \sqsubseteq \neg A$ for B' either A or any basic concept with $d \in B'^{\mathcal{I}_i}$. If $B' = A$, rule C5 of the construction of $\mathcal{T}^{\mathsf{NI}}$ yields $B \sqsubseteq \neg B \in \mathcal{T}^{\mathsf{NI}}$, and this inclusion is clearly violated by $\mathcal{I}_i$. Otherwise, $d \in B'^{\mathcal{I}_i}$. Thus the GCI $B \sqsubseteq \neg B' \in \mathcal{T}^{\mathsf{NI}}$ generated by rules C3 and C4 is violated by $\mathcal{I}_i$.

R2. Then there is a $d \in B^{\mathcal{I}_i}$ and a $B \sqsubseteq \exists r \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding a fresh element f to $\Delta^{\mathcal{I}_{i+1}}$ and (d, f) to the extension of r. This can result in the violation of negative GCIs from $\mathcal{T}^{\mathsf{NI}}$ that are of the form $\exists \mathsf{Inv}(r) \sqsubseteq \neg \exists \mathsf{Inv}(r)$, $\exists r \sqsubseteq \neg B'$ or $B' \sqsubseteq \neg \exists r$ for B' either $\exists r$ or any basic concept with $d \in B'^{\mathcal{I}_i}$. The latter two cases can be dealt with exactly as for R1. Thus assume that $\exists \mathsf{Inv}(r) \sqsubseteq \neg \exists \mathsf{Inv}(r)$ is violated. Then rule C6 has added $B \sqsubseteq \neg B$ to $\mathcal{T}^{\mathsf{NI}}$, which is violated by $\mathcal{I}_i$.

R3. Then there is a $(d, e) \in r^{\mathcal{I}_i}$ and an $r \sqsubseteq s \in \mathcal{T}$ such that $\mathcal{I}_{i+1}$ was obtained from $\mathcal{I}_i$ by adding (d, e) to the extension of s. This can result in the violation of negative GCIs from $\mathcal{T}^{\mathsf{NI}}$ that are of the form $\exists s \sqsubseteq \neg \exists s$, $\exists \mathsf{Inv}(s) \sqsubseteq \neg \exists \mathsf{Inv}(s)$ or $\exists s \sqsubseteq \neg B'$ such that $d \in B'^{\mathcal{I}_i}$, or $\exists \mathsf{Inv}(s) \sqsubseteq \neg B'$ such that $e \in B'^{\mathcal{I}_i}$. In the first two cases, C9 ensures that $\mathcal{T}^{\mathsf{NI}}$ contains $\exists r \sqsubseteq \neg \exists r$, which is violated by $\mathcal{I}_i$. In the latter two cases, C7 and C8 make $\mathcal{T}^{\mathsf{NI}}$ contain $\exists r \sqsubseteq \neg B'$ and $\exists \mathsf{Inv}(r) \sqsubseteq \neg B'$, respectively, which are both violated by $\mathcal{I}_i$. □

7.2.3 FO-rewritability in DL-Lite

We now study conjunctive query answering in DL-Lite and show that FO-rewritings of conjunctive queries with respect to DL-Lite TBoxes always exist. Thus, conjunctive query answering in DL-Lite can be delegated to a relational database system. We start by showing that the universal model plays a special role for conjunctive query answering: the certain answers to a CQ q on a DL-Lite KB $\mathcal{K}$ are identical to the answers to q on the interpretation $\mathcal{I}_{\mathcal{K}}$. Note that this is quite remarkable: while the definition of certain answers quantifies over all (infinitely many)

models of $\mathcal{K}$, it turns out that it is still sufficient to consider only one single model, which is $\mathcal{I}_\mathcal{K}$. The proof relies crucially on the fundamental Lemma 7.14.

Lemma 7.17. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a consistent DL-Lite KB and q a CQ. Then $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{I}_\mathcal{K})$.*

Proof. The "$\subseteq$" direction is clear: if $\vec{a} \notin \mathsf{ans}(q, \mathcal{I}_\mathcal{K})$, then $\vec{a} \notin \mathsf{cert}(q, \mathcal{K})$ since, by Lemma 7.15, $\mathcal{I}_\mathcal{K}$ is a model of $\mathcal{K}$.

For the "$\supseteq$" direction, assume that $\vec{a} \in \mathsf{ans}(q, \mathcal{I}_\mathcal{K})$. Then there is an $\vec{a}$-match π of q in $\mathcal{I}_\mathcal{K}$. Take any model $\mathcal{I}$ of $\mathcal{K}$. We have to show that there is an $\vec{a}$-match τ of q in $\mathcal{I}$. By Lemma 7.14, there is a homomorphism h from $\mathcal{I}_\mathcal{K}$ to $\mathcal{I}$. Define τ by setting $\tau(x) = h(\pi(x))$ for all variables x in q. Using the definitions of matches and homomorphisms, it is easy to verify that τ is an $\vec{a}$-match of q in $\mathcal{I}$, as required. □

We have already remarked that a match is nothing but a homomorphism. In summary, the "$\supseteq$" direction of Lemma 7.17 is thus a consequence of Lemma 7.14 and the fact that the composition of two homomorphisms is again a homomorphism.

As a next step, it is interesting to observe that negative concept inclusions in the TBox can make the ABox inconsistent with respect to the TBox, but otherwise have no effect on query answering. This is stated more precisely by the following lemma. Note that, if $\mathcal{K}$ is an inconsistent KB and q a CQ of arity k, then $\mathsf{cert}(q, \mathcal{K})$ is the (uninteresting) set of all k-tuples over $\mathsf{Ind}(\mathcal{A})$.

Lemma 7.18. *Let $\mathcal{K}$ be a consistent DL-Lite KB and q a CQ. Then $\mathsf{cert}(q, \mathcal{K}) = \mathsf{cert}(q, \mathcal{K}')$, where $\mathcal{K}'$ is obtained from $\mathcal{K}$ by removing from $\mathcal{T}$ all negative concept inclusions.*

Proof. "$\subseteq$". Note that $\mathcal{I}_\mathcal{K} = \mathcal{I}_{\mathcal{K}'}$ since negative concept inclusions are not used in the construction of universal models. Thus by Lemma 7.17 we have $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{K}) = \mathsf{ans}(q, \mathcal{K}') = \mathsf{cert}(q, \mathcal{K}')$. □

Recall from (the proof of) Theorem 7.8 that the non-existence of FO-rewritings is related to non-locality. For example, the query $A(x)$ turned out not to be FO-rewritable with respect to the $\mathcal{EL}$ TBox $\{\exists r.A \sqsubseteq A\}$ because there is no bound on how far the concept name A is propagated through the data, e.g., on the ABoxes $\mathcal{A}_i = \{A(a_0), r(a_1, a_0), \ldots, r(a_i, a_{i-1})\}$ for $i \geq 0$. DL-Lite is designed to avoid any such propagation and is thus "local in nature", which is responsible

for the fact that FO-rewritings of CQs with respect to DL-Lite TBoxes always exist. The locality of DL-Lite is made precise by the following lemma. For a CQ q, we define $\mathsf{size}(q)$ by analogy with the size of concepts and TBoxes: $\mathsf{size}(q)$ is the number of symbols needed to write q, counting concept and role names as one and not counting brackets.

Lemma 7.19. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a consistent DL-Lite KB, q a CQ and $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$. Then there is an $\mathcal{A}' \subseteq \mathcal{A}$ such that $|\mathsf{Ind}(\mathcal{A}')| \leq \mathsf{size}(q) \cdot (\mathsf{size}(\mathcal{T}) + 1)$ and $\vec{a} \in \mathsf{cert}(q, \mathcal{K}')$ where $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$.*

Proof. Since $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$, we have $\vec{a} \in \mathsf{ans}(q, \mathcal{I}_\mathcal{K})$ and thus there is an $\vec{a}$-match π of q in $\mathcal{I}_\mathcal{K}$. By construction, the universal model $\mathcal{I}_\mathcal{K}$ consists of an ABox part whose elements are exactly the individuals in $\mathcal{A}$ and (potentially infinite) tree-shaped parts, one rooted at each ABox individual (as an example, consider the universal model in Figure 7.2). Let I be the set of those individual names $a \in \mathsf{Ind}(\mathcal{A})$ such that π maps some variable in q to a or to a node in the tree in $\mathcal{I}_\mathcal{K}$ rooted at a. Clearly, $|I| \leq \mathsf{size}(q)$. Now extend I to J as follows: whenever $a \in I$ and r is a (potentially inverse) role in $\mathcal{T}$ such that there is an assertion $r(a, b) \in \mathcal{A}$, then choose such an $r(a, b)$ and include b in J. Clearly, $|J| \leq \mathsf{size}(q) \cdot (\mathsf{size}(\mathcal{T}) + 1)$. Let $\mathcal{A}'$ be the restriction of $\mathcal{A}$ to the individuals in J; that is, $\mathcal{A}'$ is obtained from $\mathcal{A}$ by dropping all assertions that involve an individual not in J. In the following, we show that $\vec{a} \in \mathsf{cert}(q, \mathcal{K}')$ as required.

It suffices to prove that there is an $\vec{a}$-match of q in $\mathcal{I}_{\mathcal{K}'}$. Let $\mathcal{I}_\mathcal{K}|_I^\downarrow$ be the restriction of $\mathcal{I}_\mathcal{K}$ to the elements that are either in I or located in a subtree rooted at some element of I. By choice of I, π is an $\vec{a}$-match of q in $\mathcal{I}_\mathcal{K}|_I^\downarrow$. Since the composition of two homomorphisms is a homomorphism, it thus remains to show that there is a homomorphism τ from $\mathcal{I}_\mathcal{K}|_I^\downarrow$ to $\mathcal{I}_{\mathcal{K}'}$ that is the identity on $\vec{a}$.

Let $\mathcal{I}_0, \mathcal{I}_1, \ldots$ be the interpretations from the construction of $\mathcal{I}_\mathcal{K}$; that is, $\mathcal{I}_\mathcal{K}$ is the limit of this sequence. Further, let $\mathcal{I}_i|_I^\downarrow$ be the restriction of $\mathcal{I}_i$ defined analogously to $\mathcal{I}_\mathcal{K}|_I^\downarrow$, for all $i \geq 0$. We show by induction on i that there is a homomorphism τ_i from $\mathcal{I}_i|_I^\downarrow$ to $\mathcal{I}_{\mathcal{K}'}$ such that τ_i is the identity on $\vec{a}$. In fact, we will construct these homomorphisms such that $\tau_0 \subseteq \tau_1 \subseteq \cdots$ and in the limit we obtain the desired homomorphism τ. Moreover, we construct τ_i such that the following condition is satisfied:

(∗) if $d \in C^{\mathcal{I}_i}$ with C a basic DL-Lite concept and $d \in \Delta^{\mathcal{I}_i|_I^\downarrow}$, then $\tau_i(d) \in C^{\mathcal{I}_{\mathcal{K}'}}$.

Note that (∗) does not follow from the pure existence of the homomor-

phism τ_i because the precondition says $d \in C^{\mathcal{I}_i}$ and not $d \in C^{\mathcal{I}_i|_I^{\downarrow}}$. In fact, we have extended I to J before defining $\mathcal{K}'$ precisely in order to be able to attain Property $(*)$.

For the induction start, set $\tau_0(a) = a$ for all $a \in I$. Since $\mathcal{I}_0$ is read off from $\mathcal{A}$ and $\mathcal{I}_{\mathcal{K}'}$ satisfies the restriction of $\mathcal{A}$ to individuals from I, τ_0 is a homomorphism from $\mathcal{I}_i|_I^{\downarrow}$ to $\mathcal{I}_{\mathcal{K}'}$. Clearly, τ_0 is the identity on $\vec{a}$. Moreover, it is not hard to verify that τ_0 satisfies $(*)$. For example, assume that $d \in (\exists r)^{\mathcal{I}_0}$ and $d \in \Delta^{\mathcal{I}_0|_I^{\downarrow}}$. Then $d = a$ for some $a \in I$ and $d \in (\exists r)^{\mathcal{I}_0}$ means that there is an $r(a, b) \in \mathcal{A}$. Consequently, we have chosen such an $r(a, b)$ and included b in J, thus $(a, b) \in r^{\mathcal{I}_{\mathcal{K}'}}$, which implies $\tau_0(a) \in (\exists r)^{\mathcal{I}_{\mathcal{K}'}}$ as required by $(*)$.

For the induction step, we make a case distinction according to the rule that was applied to construct $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$. Since all three cases are extremely similar, we consider only $\mathsf{R2}$ explicitly. If this rule was applied to constuct $\mathcal{I}_{i+1}$ from $\mathcal{I}_i$, then there is a $d \in B^{\mathcal{I}_i}$ and a $B \sqsubseteq \exists r \in \mathcal{T}$ such that $\Delta^{\mathcal{I}_{i+1}}$ contains a fresh element f with $(d, f) \in r^{\mathcal{I}_{i+1}}$. If $d \notin \Delta^{\mathcal{I}_i|_I^{\downarrow}}$, then neither is f and there is nothing to do. Otherwise, $(*)$ delivers $\tau_i(d) \in B^{\mathcal{I}_{\mathcal{K}'}}$. Since $\mathcal{I}_{\mathcal{K}'}$ is a model of $\mathcal{T}$, there is an $e \in \Delta^{\mathcal{I}_{\mathcal{K}'}}$ such that $(d, e) \in r^{\mathcal{I}_{\mathcal{K}'}}$. Set $\tau_{i+1} = \tau_i \cup \{f \mapsto e\}$. It is not hard to verify that τ_{i+1} is as required. In particular, $(*)$ is satisfied (also when d is the fresh element f) simply by induction hypothesis and construction of τ_{i+1}. □

Lemma 7.19 suggests a way to produce an FO-rewriting $q_{\mathcal{T}}$ for a given CQ q of arity k and DL-Lite TBox $\mathcal{T}$, by making $q_{\mathcal{T}}$ check the existence of certain sub-ABoxes of bounded size. To make this precise, let $\mathcal{T}$ be a DL-Lite TBox and q a CQ of arity k. Further, let $m = \mathsf{size}(q) \cdot (\mathsf{size}(\mathcal{T}) + 1)$ be the bound from Lemma 7.19. Fix individual names $\mathsf{Ind}_0 = \{a_1, \ldots, a_m\}$. We will consider ABoxes that only use individual names from Ind_0, contain all of $\vec{a}_0 = a_1, \ldots, a_k$ and use only concept and role names that occur in $\mathcal{T}$ or q. Such an ABox can be seen as a k-ary CQ $q_{\mathcal{A}}$ as follows:

- the variables are x_a, $a \in \mathsf{Ind}(\mathcal{A})$, where $x_{a_1}, \ldots, x_{a_k}$ are the answer variables and all other variables are quantified;
- every concept assertion $A(a)$ in $\mathcal{A}$ gives rise to a concept atom $A(x_a)$ in $q_{\mathcal{A}}$ and every role assertion $r(a, b) \in \mathcal{A}$ gives rise to a role atom $r(x_a, x_b)$ in $q_{\mathcal{A}}$; these are the only atoms.

Define $q_{\mathcal{T}}$ to be the disjunction of all CQs $q_{\mathcal{A}}$ such that $\mathcal{A}$ is an ABox as above and $\vec{a}_0 \in \mathsf{cert}(q, \mathcal{K})$, where $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

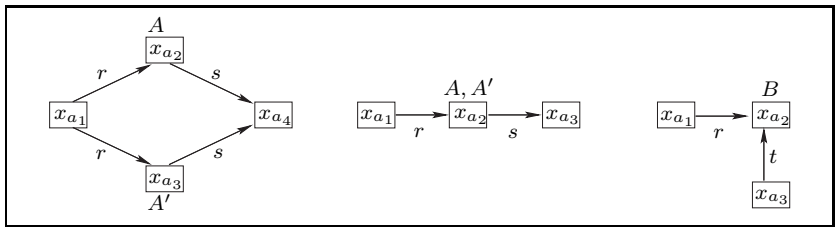

Fig. 7.3. Some disjuncts of $q_{\mathcal{T}}$.

As an example, consider the following TBox and CQ:

$$\begin{aligned} \mathcal{T} &= \{\exists t^- \sqsubseteq A,\ B \sqsubseteq \exists s\}, \\ q(x) &= r(x, y_1) \wedge r(x, y_2) \wedge A(y_1) \wedge A'(y_2) \wedge s(y_1, z) \wedge s(y_2, z). \end{aligned}$$

In Figure 7.3, we show three example disjuncts in $q_{\mathcal{T}}$. The left-most query is simply q itself, up to variable renaming. The middle query must always be a disjunct of $q_{\mathcal{T}}$, independently of what $\mathcal{T}$ actually is. In contrast, the presence of the right-most query as a disjunct in $q_{\mathcal{T}}$ does depend on $\mathcal{T}$. As a small exercise, the reader might want to convert this query into an ABox by replacing each variable x_{a_i} with an individual name a_i, construct the universal model $\mathcal{I}_{\mathcal{K}}$ of the resulting KB $\mathcal{K}$ and then verify that $a_1 \in \mathsf{ans}(q, \mathcal{I}_{\mathcal{K}})$, thus $a_1 \in \mathsf{cert}(q, \mathcal{K})$. Note that disjuncts of $q_{\mathcal{T}}$ might contain a larger number of variables than the original query, one example being obtained by replacing in $q(x)$ the atom $A(y_1)$ with $t(u, y_1)$.

It is easy to establish that $q_{\mathcal{T}}$ is *complete* as an FO-rewriting, meaning the following.

Lemma 7.20 (Completeness)**.** *For any consistent KB* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, $\mathsf{cert}(q, \mathcal{K}) \subseteq \mathsf{ans}(q_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}})$.

Proof. Let $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$. By Lemma 7.19, we find an $\mathcal{A}' \subseteq \mathcal{A}$ of size at most $\mathsf{size}(q) \cdot (\mathsf{size}(\mathcal{T}) + 1)$ and with $\vec{a} \in \mathsf{cert}(q, \mathcal{K}')$ for $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$. By renaming the individual names, we obtain from $\mathcal{A}'$ an ABox $\mathcal{B}$ that uses only individual names from Ind_0 and such that $\vec{a}_0 \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{B}))$. Then $q_{\mathcal{B}}$ must be a disjunct of $q_{\mathcal{T}}$, and it can be verified that we find an $\vec{a}$-match π of $q_{\mathcal{B}}$ in $\mathcal{I}_{\mathcal{A}}$ by setting $\pi(x_a) = b$ if b is the individual name that was renamed to a when constructing $\mathcal{B}$. Consequently $\vec{a} \in \mathsf{ans}(q_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}})$ as required. □

However, we also wish to show that $q_{\mathcal{T}}$ is sound, which means that

the converse of Lemma 7.20 holds. As a technical preliminary, we first observe that query answers are preserved under taking the homomorphic pre-image of an ABox (and the answer). Homomorphisms between ABoxes are defined in the obvious way: $h : \mathsf{Ind}(\mathcal{A}_1) \to \mathsf{Ind}(\mathcal{A}_2)$ is a *homomorphism from ABox* $\mathcal{A}_1$ *to ABox* $\mathcal{A}_2$ if the following conditions are satisfied:

(i) $A(a) \in \mathcal{A}_1$ implies $A(h(a)) \in \mathcal{A}_2$;
(ii) $r(a, b) \in \mathcal{A}_1$ implies $r(h(a), h(b)) \in \mathcal{A}_2$.

For later use, we state the lemma for KBs formulated in $\mathcal{ALCI}$, of which DL-Lite is a very moderate fragment. It actually holds true for all ontology languages that do not admit nominals.

Lemma 7.21. *Let* $\mathcal{K}_i = (\mathcal{T}, \mathcal{A}_i)$ *for* $i \in \{1, 2\}$ *be an* $\mathcal{ALCI}$*-KB,* q *a CQ and* h *a homomorphism from* $\mathcal{A}_1$ *to* $\mathcal{A}_2$*. Then* $\vec{a} \in \mathsf{cert}(q, \mathcal{K}_1)$ *implies* $h(\vec{a}) \in \mathsf{cert}(q, \mathcal{K}_2)$*.*

Proof. We show that $h(\vec{a}) \notin \mathsf{cert}(q, \mathcal{K}_2)$ implies $\vec{a} \notin \mathsf{cert}(q, \mathcal{K}_1)$. If $h(\vec{a}) \notin \mathsf{cert}(q, \mathcal{K}_2)$, then there is a model $\mathcal{I}$ of $\mathcal{K}_2$ such that q has no $h(\vec{a})$-match in $\mathcal{I}$. Define an interpretation $\mathcal{J}$ by starting with $\mathcal{I}$ and changing the interpretation of all $a \in \mathsf{Ind}(\mathcal{A}_1)$ by setting $a^{\mathcal{J}} = h(a)^{\mathcal{I}}$. Since the interpretation of TBoxes is independent of individual names, $\mathcal{J}$ is a model of $\mathcal{T}$. Using conditions (i) and (ii) in the definition of ABox homomorphisms, it is straightforward to verify that $\mathcal{J}$ is also a model of $\mathcal{A}_1$. For example, $A(a) \in \mathcal{A}_1$ implies $A(h(a)) \in \mathcal{A}_2$ and thus $h(a)^{\mathcal{I}} \in A^{\mathcal{I}}$ since $\mathcal{I}$ is a model of $\mathcal{A}_2$; consequently $a^{\mathcal{J}} \in A^{\mathcal{J}}$. It thus remains to observe that there is no $\vec{a}$-match of q in $\mathcal{J}$ as this implies $\vec{a} \notin \mathsf{cert}(q, \mathcal{K}_1)$ as desired. In fact, any $\vec{a}$-match of q in $\mathcal{J}$ is, by definition of $\mathcal{J}$, also an $h(\vec{a})$-match of q in $\mathcal{I}$, contradicting the fact that no such match exists. □

Note that the (very simple) proof of Lemma 7.21 crucially relies on not making the unique name assumption. Lemma 7.21 also holds with the UNA, but is a bit more difficult to prove. We can now establish soundness of the FO-rewriting $q_{\mathcal{T}}$ of the CQ q relative to the TBox $\mathcal{T}$.

Lemma 7.22 (Soundness)**.** *For any KB* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$*,* $\mathsf{ans}(q_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}}) \subseteq \mathsf{cert}(q, \mathcal{K})$*.*

Proof. Let $\vec{a} \in \mathsf{ans}(q_{\mathcal{T}}, \mathcal{I}_{\mathcal{A}})$. Then there is a disjunct $q_{\mathcal{B}}$ of $q_{\mathcal{T}}$ such that $\vec{a} \in \mathsf{ans}(q_{\mathcal{B}}, \mathcal{I}_{\mathcal{A}})$. Consequently, there is an $\vec{a}$-match π of $q_{\mathcal{B}}$ in $\mathcal{I}_{\mathcal{A}}$. Define a map $h : \mathsf{Ind}(\mathcal{B}) \to \mathsf{Ind}(\mathcal{A})$ by setting $h(b) = a$ if $\pi(x_b) = a$. It

can be verified that h is a homomorphism from $\mathcal{B}$ to $\mathcal{A}$ with $h(\vec{a}_0) = \vec{a}$. Since $\vec{a}_0 \in \mathsf{cert}(q, (\mathcal{T}, \mathcal{B}))$, Lemma 7.21 thus yields that $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$ as required. □

Summing up, we have established the following.

Theorem 7.23. *For every DL-Lite TBox $\mathcal{T}$ and CQ q, there exists an FO-rewriting $q_{\mathcal{T}}$.*

Note that, by definition of FO-rewritability, the FO-rewriting $q_{\mathcal{T}}$ is only guaranteed to deliver the desired answers when executed on an ABox $\mathcal{A}$ such that the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent. Before answering any queries, we would thus like to find out whether $\mathcal{K}$ is consistent and notify the user when this is not the case. Fortunately, the required check can be implemented by querying and thus also delegated to the relational database system that stores $\mathcal{A}$.

Theorem 7.24. *Let $\mathcal{T}$ be a DL-Lite TBox and $\mathcal{T}'$ be obtained from $\mathcal{T}$ by removing all negative concept inclusions. Then there is a finite set Q of Boolean CQs such that $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent if and only if $\mathcal{K}' \not\models q$ for all $q \in Q$, where $\mathcal{K}' = (\mathcal{T}', \mathcal{A})$.*

Proof. We construct the desired set of Boolean CQs Q by including one query for every negative concept inclusion $B_1 \sqsubseteq \neg B_2$ in $\mathcal{T}$. For example, if $\mathcal{T}$ contains $A \sqsubseteq \neg B$, then we include $\exists x\,(A(x) \wedge B(x))$ in Q, and if $\mathcal{T}$ contains $\exists r \sqsubseteq \neg \exists s^-$ then we include $\exists x \exists y \exists z\,(r(x, y) \wedge s(z, x))$. It remains to show that Q is as required.

(if) Assume $\mathcal{K}' \not\models q$ for all $q \in Q$. Since $\mathcal{K}'$ contains no negative inclusions, it is consistent. Thus, $\mathcal{I}_{\mathcal{K}'}$ is a model of $\mathcal{K}'$. Moreover, $\mathcal{I}_{\mathcal{K}'} \not\models q$ for all $q \in Q$ by Lemma 7.17. By definition of the queries in Q, it follows that $\mathcal{I}_{\mathcal{K}'}$ satisfies all negative concept inclusions in $\mathcal{T}$. Thus $\mathcal{I}_{\mathcal{K}'}$ is a model of $\mathcal{K}$ and $\mathcal{K}$ is consistent.

(only if) If $\mathcal{K}$ is consistent, then $\mathcal{I}_{\mathcal{K}}$ is a model of $\mathcal{K}$ and thus all negative concept inclusions from $\mathcal{T}$ are satisfied in $\mathcal{I}_{\mathcal{K}}$. Consequently, $\mathcal{I}_{\mathcal{K}} \not\models q$ for all $q \in Q$. Since $\mathcal{I}_{\mathcal{K}}$ is a model $\mathcal{K}'$, we obtain $\mathcal{K}' \not\models q$ for all $q \in Q$. □

We remark that the construction of FO-rewritings $q_{\mathcal{T}}$ that underlies Theorem 7.23 is not yet effective since it requires us to decide, given a KB $\mathcal{K}$, CQ q and tuple $\vec{a}$, whether $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$. We briefly discuss how this can be done, but emphasise that, from a practical perspective, the described construction of $q_{\mathcal{T}}$ is suboptimal anyway since it often results in unnecessarily large rewritings. This is discussed further after the proof of Theorem 7.25.

Theorem 7.25. *Given a DL-Lite KB $\mathcal{K}$, CQ q and tuple of individual names $\vec{a}$, it is decidable whether $\vec{a} \in \mathsf{cert}(q, \mathcal{K})$.*

Sketch of Proof. Given $\mathcal{K}$, q and $\vec{a}$, we construct a finite initial part $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$ of $\mathcal{I}_\mathcal{K}$ and then check whether there is an $\vec{a}$-match of q in $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$ by considering all candidates, that is, all mappings from the variables in q to the elements of $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$. More precisely, $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$ is the restriction of $\mathcal{I}_\mathcal{K}$ to the domain elements d that are on level at most $\mathsf{size}(q) + \mathsf{size}(\mathcal{T})$, meaning that d can be reached from an ABox individual by travelling along a role path of length at most $\mathsf{size}(q) + \mathsf{size}(\mathcal{T})$. It thus remains to argue that, if there is an $\vec{a}$-match π of q in $\mathcal{I}_\mathcal{K}$, then there is such a match in $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$. It suffices to show this for all queries q that are connected (when viewed as a graph where the variables are the nodes and the role atoms the edges) since, for disconnected queries, we can treat all maximal connected components separately.

Thus assume that q is connected. The *depth* of a match is the minimum of the levels of all elements in the range of the match. Assume without loss of generality that π is of minimal depth. We aim to show that the depth of π is bounded by $\mathsf{size}(\mathcal{T})$ because then π maps all variables to elements of level $\leq \mathsf{size}(q) + \mathsf{size}(\mathcal{T})$ due to connectedness; thus π falls within $\mathcal{I}_\mathcal{K}^{\mathsf{ini}}$ as required.

Let d be an element of smallest level in the range of π. Since q is connected and d is of smallest level, π maps all variables in q to the subtree of $\mathcal{I}_\mathcal{K}$ rooted at d. Assume to the contrary of what we want to show that the level of d exceeds $\mathsf{size}(\mathcal{T})$. Then q must be a Boolean query. By construction of $\mathcal{I}_\mathcal{K}$, there is a unique path $a = d_0, \dots, d_k = d$ from an ABox individual a to d. For $0 \leq i < k$, let C_i be the basic DL-Lite concept such that $d_i \in C_i^{\mathcal{I}_\mathcal{K}}$ and d_{i+1} was generated by an application of $\mathsf{R1}$ or $\mathsf{R2}$ to $d_i \in C_i^{\mathcal{I}_\mathcal{K}}$. Since the number of basic concepts is bounded by $\mathsf{size}(\mathcal{T})$, we must have $C_j = C_\ell$ for some j, ℓ with $0 \leq j < \ell < k$. The construction of $\mathcal{I}_\mathcal{K}$ ensures that the subtree rooted at d_j is identical to the subtree rooted at d_ℓ. Thus there must be an element e below d_j that, like d, was generated by an application of $\mathsf{R1}$ or $\mathsf{R2}$ to C_{k-1} but whose level is smaller than that of d (since the level of d_j is smaller than that of d_ℓ). The subtree rooted at e must be identical to that rooted at d and thus there is a match of q in $\mathcal{I}_\mathcal{K}$ that involves e, in contradiction to the minimal depth of π. □

As mentioned above, the rewriting strategy presented here is not optimal from a practical perspective. In fact, it produces exponentially large rewritings in the *best case*, in the size of both the TBox and the

query. We have chosen this form of rewriting only because it is conceptually simple and underlines the importance of locality (in the sense of Gaifman) for the existence of FO-rewritings and the design of DL-Lite.

Using results from the area of circuit complexity, it can be proved that exponentially-sized rewritings cannot be avoided in general. However, by constructing rewritings in a more goal-directed way and making use of various existing optimisation techniques, in practice it is often possible to find rewritings that are of reasonable size. While giving full details is beyond the scope of this chapter, we briefly mention that *backwards chaining* provides a more goal-directed approach to constructing rewritings. The idea is to (repeatedly and in all possible ways) replace atoms in the query by other atoms whenever $\mathcal{T}$ ensures that the latter imply the former. This approach is complicated by the fact that, additionally, it can sometimes be necessary to identify variables in the query. As an example, consider the query $q(x)$ and TBox $\mathcal{T}$ whose rewritings are displayed in Figure 7.3. The left-most rewriting is the original query and from it one reaches the middle rewriting by identifying the variables x_{a_2} and x_{a_3}. One backwards chaining step then replaces $A(x_{a_2})$ with $t(y, x_{a_2})$. Another backwards chaining step replaces $s(x_{a_2}, x_{a_3})$ with $B(x_{a_2})$, producing the right-most rewriting. Note that this last step is not possible without prior identification of x_{a_2} and x_{a_3}.

7.3 Datalog-rewritability in $\mathcal{EL}$ and $\mathcal{ELI}$

We now consider query answering in the description logics $\mathcal{EL}$ and $\mathcal{ELI}$ that, unlike DL-Lite, are able to express recursive queries. As a paradigmatic example, recall from Lemma 7.19 that $A(x)$ is not FO-rewritable with respect to the $\mathcal{EL}$ TBox $\mathcal{T} = \{\exists r.A \sqsubseteq A\}$ because the concept name A has to be propagated unboundedly far through the data. Since SQL provides only limited support of recursion, the query rewriting techniques for DL-Lite which we have developed in Section 7.2 cannot be adapted to $\mathcal{EL}$ and beyond.[2] One possible way around this problem is to admit only acyclic $\mathcal{EL}$ TBoxes (see Definition 2.9), which prevents unbounded recursion. In fact, it can be shown that conjunctive queries are always FO-rewritable with respect to acyclic $\mathcal{EL}$ TBoxes. If acyclic TBoxes are not sufficient, the only choice is to replace SQL as the target language of rewriting with a query language that admits recursion,

[2] SQL allows no recursion in Versions 1 and 2, and linear recursion from Version 3 on. The latter is not strong enough to guarantee the existence of rewritings in the context of $\mathcal{EL}$.

most notably Datalog. While Datalog engines are not as mainstream as SQL databases, there is still a substantial number of highly optimised such systems available, making Datalog a very suitable target for rewriting. In this section, we are concerned with rewriting into Datalog. For simplicity, we shall only consider conjunctive queries of the form $A(x)$, which from now on we call an *atomic query* (AQ). A few remarks on how to extend the presented results to full conjunctive queries are given at the end of the section.

7.3.1 Fundamentals of Datalog

Datalog is a simple and appealing query language with a rule-based syntax. A *Datalog rule* ρ has the form

$$R_0(\mathbf{x}_0) \leftarrow R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n),$$

with $n > 0$ and where each R_i is a relation symbol with an associated arity and each $\mathbf{x}_i$ is a tuple of variables whose length coincides with the arity of R_i. We refer to $R_0(\mathbf{x}_0)$ as the *head* of ρ and to $R_1(\mathbf{x}_1) \wedge \cdots \wedge R_n(\mathbf{x}_n)$ as the *body*. Every variable that occurs in the head of a rule is required to also occur in the body. A *Datalog program* Π is a finite set of Datalog rules, for example

$$\begin{array}{rcl} X_A(x) & \leftarrow & A(x), \\ X_A(x) & \leftarrow & r(x,y) \wedge X_A(y), \\ \mathsf{goal}(x) & \leftarrow & X_A(x). \end{array}$$

Relation symbols that occur in the head of at least one rule are *intensional* or *IDB* relations (X_A and goal in the above program) and all remaining relation symbols are *extensional* or *EDB* relations (A and r in the above program). Intuitively, the EDB relations are those relations that are allowed to occur in the data while the IDB relations are additional relations that only help in defining the query. We assume that there is a selected *goal relation* goal that does not occur in rule bodies. The *arity of* Π is the arity of the goal relation.

To fit within the framework of Description Logic, we assume that all relations (both EDB and IDB) are unary or binary, identifying unary relations with concept names and binary relations with role names. Given that we restrict ourselves to atomic query answering, we also assume that the goal relation is unary and, consequently, so are Datalog programs. Observe that the bodies of Datalog rules are actually conjunctive

queries, and the heads are CQs that consist of a single atom. An interpretation $\mathcal{I}$ *satisfies* a Datalog rule ρ if every match of the (CQ that is the) body of ρ is also a match of the head. For example, the rule

$$r(x, y) \leftarrow r(x, z) \wedge r(z, y)$$

is satisfied in $\mathcal{I}$ if $r^{\mathcal{I}}$ is transitive.

Let Π be a Datalog program and $\mathcal{I}$ an interpretation that represents a database and in which the extension of all IDB relations is empty. Intuitively, an element $d \in \Delta^{\mathcal{I}}$ is an *answer* to Π on $\mathcal{I}$ if exhaustive application of the rules in Π to $\mathcal{I}$ results in d being in the extension of goal. Formally, we require that, for every interpretation $\mathcal{J}$ which can be obtained from $\mathcal{I}$ by extending the interpretation of IDB relations (concept and role names that occur in a rule head) and which satisfies all rules in Π, we have $d \in \mathsf{goal}^{\mathcal{J}}$. We use $\mathsf{ans}(\Pi, \mathcal{I})$ to denote the set of all answers to Π in $\mathcal{I}$.

Now that we have defined the syntax and semantics of Datalog, let us use this query language as a target for rewriting atomic queries with respect to TBoxes. Intuitively, it should be clear that the three-rule example Datalog program above is a rewriting of the atomic query $A(x)$ with respect to the TBox $\{\exists r.A \sqsubseteq A\}$. The formal definition is as follows.

Definition 7.26. Let $\mathcal{T}$ be a TBox and q a conjunctive query. A Datalog program Π is a *Datalog-rewriting* of q with respect to $\mathcal{T}$ if, for all ABoxes $\mathcal{A}$, we have $\mathsf{cert}(q, \mathcal{K}) = \mathsf{ans}(\Pi, \mathcal{I}_{\mathcal{A}})$, where $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

We have not required consistency of $\mathcal{K}$ in the above definition only because in this section we work with description logics that cannot produce inconsistencies. When inconsistency can occur, one would require $\mathcal{K}$ to be consistent, exactly as in Definition 7.7.

The following is an interesting first observation that relates Datalog-rewritings to FO-rewritings.

Lemma 7.27. *Let $A_0(x)$ be an AQ and $\mathcal{T}$ an $\mathcal{ALCI}$ TBox. If $A_0(x)$ is FO-rewritable with respect to $\mathcal{T}$, then $A_0(x)$ is Datalog-rewritable with respect to $\mathcal{T}$.*

Sketch of proof. An FO query q is *preserved under homomorphisms* if it satisfies the following property: if $\vec{d} \in \mathsf{ans}(q, \mathcal{I})$ and h is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$, then $h(\vec{d}) \in \mathsf{ans}(q, \mathcal{J})$. A classical result in model theory states that an FO query is preserved under homomorphisms if and only if it is equivalent to a positive existential FO query, that is, a query

composed only of conjunction, disjunction and existential quantifiers. This result was lifted to the class of finite interpretations by Rossman. As a consequence of Lemma 7.21, an FO-rewriting $q(x)$ of $A_0(x)$ with respect to $\mathcal{T}$ is preserved under homomorphisms on the class of finite interpretations: if h is a homomorphism from $\mathcal{I}$ to $\mathcal{J}$ with $\mathcal{I}$ and $\mathcal{J}$ finite, then $d \in \mathsf{ans}(q, \mathcal{I})$ implies $d \in \mathsf{cert}(A_0(x), \mathcal{K}_\mathcal{I})$ implies $h(d) \in \mathsf{cert}(A_0(x), \mathcal{K}_\mathcal{J})$ implies $h(d) \in \mathsf{ans}(q, \mathcal{J})$, where $\mathcal{K}_\mathcal{I} = (\mathcal{T}, \mathcal{A}_\mathcal{I})$ and $\mathcal{A}_\mathcal{I}$ is the ABox such that $\mathcal{I}_{\mathcal{A}_\mathcal{I}} = \mathcal{I}$, and likewise for $\mathcal{K}_\mathcal{J}$. Consequently, a rewriting $q(x)$ of $A_0(x)$ with respect to $\mathcal{T}$ is equivalent to a positive existential FO query, thus also to an FO query of the form $\bigvee_i q_i(x)$, where each $q_i(x)$ is a CQ; such a disjunction is commonly called a *union of conjunctive queries* (UCQ). Finally, a UCQ $\bigvee_i q_i(x)$ is equivalent to the Datalog program that consists of the rules $\mathsf{goal}(x) \leftarrow q_i(x)$. □

7.3.2 Datalog-rewritings in $\mathcal{ELI}$

We show that Datalog-rewritings of atomic queries with respect to $\mathcal{ELI}$ TBoxes always exist, and how they can be constructed. The main challenge is to deal with existential restrictions on the right-hand sides of concept inclusions in the TBox. In fact, an $\mathcal{ELI}$ TBox in which no such restrictions occur is a notational variation of a Datalog program. Here, we show how to deal with the general case.

Let $\mathcal{T}$ be an $\mathcal{ELI}$ TBox and $A_0(x)$ an atomic query to be rewritten. We introduce a fresh concept name X_C for every $C \in \mathsf{sub}(\mathcal{T})$ to serve as an IDB relation in Π. The rules of Π are now as follows:

(i) for every concept name $A \in \mathsf{sub}(\mathcal{T})$, the rule $X_A(x) \leftarrow A(x)$;

(ii) for every $\exists r.C \in \mathsf{sub}(\mathcal{T})$, the rule $X_{\exists r.C}(x) \leftarrow r(x, y) \wedge X_C(y)$;

(iii) for every $C \in \mathsf{sub}(\mathcal{T})$ and every subset $\Gamma \subseteq \mathsf{sub}(\mathcal{T})$ such that $\mathcal{T} \models \bigsqcap \Gamma \sqsubseteq C$, the rule $X_C(x) \leftarrow \bigwedge_{D \in \Gamma} X_D(x)$;

(iv) the rule $\mathsf{goal}(x) \leftarrow X_{A_0}(x)$.

For readers who also expected rules of the form $X_{C \sqcap D} \leftarrow X_C \wedge X_D$, we note that these are just a special case of (iii). It is the rules of kind (iii) that deal with existential restrictions on the right-hand side. As a very simple example, consider the atomic query $A_0(x)$ and the TBox $\mathcal{T} = \{A \sqsubseteq \exists r.B, \exists r.B \sqsubseteq A_0\}$. For $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{A} = \{A(a)\}$, we have $a \in \mathsf{cert}(A_0(x), \mathcal{K})$. The Datalog-rewriting Π of A_0 with respect

to $\mathcal{T}$ contains, among others, the following rules:

$$\begin{aligned} X_A(x_0) &\leftarrow A(x_0), \\ X_{A_0}(x) &\leftarrow X_A(x), \\ \mathsf{goal}(x) &\leftarrow X_{A_0}(x), \end{aligned}$$

and thus $a \in \mathsf{ans}(\Pi, \mathcal{I}_\mathcal{A})$. The middle rule is of kind (iii), and intuitively it cuts short the existential restriction $\exists r.B$ in $\mathcal{T}$. Since such shortcuts are not always obvious (which is related to the fact that subsumption in $\mathcal{ELI}$ is ExpTime-complete), we use an exhaustive approach and consider all possible conjunctions of subconcepts in (iii).

Lemma 7.28. *Π is a Datalog-rewriting of $A_0(x)$ with respect to $\mathcal{T}$.*

Proof. We have to show that, for all ABoxes $\mathcal{A}$, $\mathsf{cert}(A_0(x), \mathcal{K}) = \mathsf{ans}(\Pi, \mathcal{I}_\mathcal{A})$, where $\mathcal{K} = (\mathcal{T}, \mathcal{A})$.

"$\subseteq$". Let $a_0 \notin \mathsf{ans}(\Pi, \mathcal{I}_\mathcal{A})$. Then there is an extension $\mathcal{J}$ of $\mathcal{I}_\mathcal{A}$ to the IDB relations in Π such that $\mathcal{J}$ satisfies all rules of Π and $a_0 \notin \mathsf{goal}^\mathcal{J}$. For every $a \in \mathsf{Ind}(\mathcal{A})$, let Γ_a be the set of all concepts $C \in \mathsf{sub}(\mathcal{T})$ such that $a \in X_C^\mathcal{J}$. Due to the rules of Π of kind (iii), Γ_a is closed under consequence; that is, if $\mathcal{T} \models \bigsqcap \Gamma_a \sqsubseteq C$ for some $C \in \mathsf{sub}(\mathcal{T})$, then $C \in \Gamma_a$. By Lemma 6.31, we find a model $\mathcal{I}_a$ of $\mathcal{T}$ with $a \in \Delta^{\mathcal{I}_a}$ and such that for all $C \in \mathsf{sub}(\mathcal{T})$, we have $\mathcal{T} \models \bigsqcap \Gamma_a \sqsubseteq C$ if and only if $a \in C^{\mathcal{I}_a}$; since Γ_a is closed under consequence, this means $C \in \Gamma_a$ if and only if $a \in C^{\mathcal{I}_a}$. Assume that each $\mathcal{I}_a$ shares only the element a with $\mathcal{I}_\mathcal{A}$ and that $\Delta^{\mathcal{I}_a} \cap \Delta^{\mathcal{I}_b} = \emptyset$ whenever $a \neq b$.

Define the interpretation $\mathcal{I}$ by first taking the disjoint union of all the interpretations $\mathcal{I}_a$ and then adding (a, b) to $r^\mathcal{I}$ whenever $r(a, b) \in \mathcal{A}$; for all individual names a, set $a^\mathcal{I} = a$. We show the following:

($*$) $d \in C^{\mathcal{I}_a}$ if and only if $d \in C^\mathcal{I}$ for all $C \in \mathsf{sub}(\mathcal{T})$, $a \in \mathsf{Ind}(\mathcal{A})$ and $d \in \Delta^{\mathcal{I}_a}$.

The proof is by induction on the structure of C. The case of concept names and conjunction is straightforward, so we consider only existential restrictions. Here, the "only if" direction is also easy, so we concentrate on "if". Assume that $d \in (\exists r.C)^\mathcal{I}$ with $d \in \Delta^{\mathcal{I}_a}$. Then there is a $(d, e) \in r^\mathcal{I}$ with $e \in C^\mathcal{I}$. By construction of $\mathcal{I}$, we have $(d, e) \in r^{\mathcal{I}_a}$ or there is a $b \in \mathsf{Ind}(\mathcal{A})$ such that $(d, e) = (a, b)$ and $r(a, b) \in \mathcal{A}$. In the former case, the induction hypothesis yields $e \in C^{\mathcal{I}_a}$ and thus $d \in (\exists r.C)^{\mathcal{I}_a}$ as required. Thus assume $(d, e) = (a, b)$ and $r(a, b) \in \mathcal{A}$. From $e \in C^\mathcal{I}$ and the induction hypothesis, we obtain $b \in C^{\mathcal{I}_b}$ and thus $C \in \Gamma_b$ by choice

of $\mathcal{I}_b$ and $b \in X_C^{\mathcal{J}}$ by choice of Γ_b. Since $r(a,b) \in \mathcal{A}$ yields $(a,b) \in r^{\mathcal{J}}$ and the rules in Π of kind (ii) are satisfied in $\mathcal{J}$, we have $a \in X_{\exists r.C}^{\mathcal{J}}$, thus $\exists r.C \in \Gamma_a$ and $d \in (\exists r.C)^{\mathcal{I}_a}$, as required.

As a consequence of $(*)$ and since each $\mathcal{I}_a$ is a model of $\mathcal{T}$, $\mathcal{I}$ is also a model of $\mathcal{T}$. By construction, it satisfies all role assertions in $\mathcal{A}$. Concept assertions are also satisfied: $A(a) \in \mathcal{A}$ implies $a \in X_A^{\mathcal{J}}$ since the rules in Π of kind (i) are satisfied, thus $a \in A^{\mathcal{I}_a}$ and $(*)$ yields $a \in A^{\mathcal{I}}$. Finally, by the rules of kind (iv), $a_0 \notin \mathsf{goal}^{\mathcal{J}}$ implies $a_0 \notin X_{A_0}^{\mathcal{J}}$, and consequently $a_0 \notin A_0^{\mathcal{I}}$. We have thus shown that $a_0 \notin \mathsf{cert}(A_0(x), \mathcal{K})$.

"$\supseteq$". Let $a_0 \notin \mathsf{cert}(A_0(x), \mathcal{K})$. Then there is a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{K}$ such that $a_0^{\mathcal{I}} \notin A_0^{\mathcal{I}}$. Let $\mathcal{J}$ be the extension of $\mathcal{I}_{\mathcal{A}}$ to the IDB relations in Π by setting $X_C^{\mathcal{J}} = \{a \in \mathsf{Ind}(\mathcal{A}) \mid a^{\mathcal{I}} \in C^{\mathcal{I}}\}$ for every IDB relation of the form X_C and $\mathsf{goal}^{\mathcal{J}} = X_{A_0}^{\mathcal{J}}$. It can be verified that the extended $\mathcal{J}$ satisfies all rules in Π and that $a_0 \notin \mathsf{goal}^{\mathcal{J}}$, and thus $a_0 \notin \mathsf{ans}(\Pi, \mathcal{I}_{\mathcal{A}})$. □

We have thus established the following result.

Theorem 7.29. *For every atomic query $A_0(x)$ and $\mathcal{ELI}$ TBox $\mathcal{T}$, there is a Datalog-rewriting Π.*

Theorem 7.29 is also true when the atomic queries A_0 are replaced with conjunctive queries q, but then the construction of Π becomes more complicated. The intuitive reason is that the existentially quantified part of a conjunctive query q can (fully or partially) be matched to elements that are generated by existential restrictions in $\mathcal{T}$, which we have "cut short" in the construction of Π as explained above. This means that it does not suffice to include in Π the rule $\mathsf{goal}(\vec{x}) \leftarrow q(\vec{x})$, but instead we have to "dissect" q according to which parts of it are matched in the ABox, and which parts are matched (implicitly!) inside the shortcuts, and then reflect this dissection in the rules. Of course, there is more than one possible such dissection of q, and all of them have to be considered.

Note that the Datalog-rewriting Π constructed above is of size exponential in the size of $\mathcal{T}$, due to the rules of kind (iii). In fact, it is known that Datalog-rewritings of polynomial size do not exist unless a standard complexity-theoretic assumption fails.[3] Of course, it is nevertheless possible to improve the presented construction of Π to make it shorter in many cases.

[3] The assumption is that ExpTime $\not\subseteq$ coNP/Poly, where the latter is the non-uniform version of the complexity class coNP, commonly defined via Turing machines with *advice*; readers are referred to complexity theory textbooks for details.

7.3.3 Short Datalog-rewritings in $\mathcal{EL}$

We now consider the case of $\mathcal{EL}$ TBoxes and refine the construction of Datalog-rewritings given above so that the resulting program is of only polynomial size. Thus, let $\mathcal{T}$ be an $\mathcal{EL}$ TBox and $A_0(x)$ an atomic query. We again use a concept name X_C for every $C \in \mathsf{sub}(\mathcal{T})$ as an IDB relation. The rules of Π are:

(i) for every concept name $A \in \mathsf{sub}(\mathcal{T})$, the rule $X_A(x) \leftarrow A(x)$;
(ii) for every $C \sqcap D \in \mathsf{sub}(\mathcal{T})$, the rules $X_{C\sqcap D}(x) \leftarrow X_C(x) \wedge X_D(x)$, $X_C(x) \leftarrow X_{C\sqcap D}(x)$ and $X_D(x) \leftarrow X_{C\sqcap D}(x)$;
(iii) for every $\exists r.C \in \mathsf{sub}(\mathcal{T})$, the rule $X_{\exists r.C}(x) \leftarrow r(x,y) \wedge X_C(y)$;
(iv) for all $C, D \in \mathsf{sub}(\mathcal{T})$ with $\mathcal{T} \models C \sqsubseteq D$, the rule $X_D(x) \leftarrow X_C(x)$;
(v) the rule $\mathsf{goal}(x) \leftarrow X_{A_0}(x)$.

Note that the former rules of kind (iii) have been replaced by what are now rules of kinds (ii) and (iv).

Lemma 7.30. *Π is a Datalog-rewriting of $A_0(x)$ with respect to $\mathcal{T}$.*

Proof. An analysis of the proof of Lemma 7.28 shows that $\mathsf{cert}(A_0(x), \mathcal{K}) \supseteq \mathsf{ans}(\Pi, \mathcal{I}_\mathcal{A})$ is still straightforward to establish and that, in the converse direction, the only step that uses the former Datalog rules of kind (iii) which have been removed in the new translation is to show the following: if $\mathcal{J}$ is an extension of $\mathcal{I}_\mathcal{A}$ to the IDB relations in Π that satisfies all rules of Π, then for all $a \in \mathsf{Ind}(\mathcal{A})$ there is an interpretation $\mathcal{I}_a$ such that $a \in \Delta^{\mathcal{I}_a}$, and for all $C \in \mathsf{sub}(\mathcal{T})$ we have $a \in C^{\mathcal{I}_a}$ if and only if $C \in \Gamma_a$, where $\Gamma_a = \{C \mid a \in X_C^\mathcal{J}\}$. We show that, when $\mathcal{T}$ is formulated in $\mathcal{EL}$, this can be established using the new rules.

Let $\exists r_1.C_1, \dots, \exists r_n.C_n$ be the existential restrictions in Γ_a. By Lemma 6.31, for $1 \leq i \leq n$ we find a model $\mathcal{I}_i$ of $\mathcal{T}$ and a $d_i \in \Delta^{\mathcal{I}_i}$ such that, for all $C \in \mathsf{sub}(\mathcal{T})$, we have $\mathcal{T} \models C_i \sqsubseteq C$ if and only if $d_i \in C^{\mathcal{I}_i}$. Assume without loss of generality that none of the $\Delta^{\mathcal{I}_i}$ contains a and that $\Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} = \emptyset$ whenever $i \neq j$. Define the interpretation $\mathcal{I}_a$ by first taking the disjoint union of all interpretations $\mathcal{I}_i$ and then adding a as a fresh root; that is, a is added to $\Delta^{\mathcal{I}_a}$ and (a, d_i) is added to $r_i^{\mathcal{I}_a}$ for $1 \leq i \leq k$. Also, add a to $A^{\mathcal{I}_a}$ whenever $X_A \in \Gamma_a$.

It remains to show that $\mathcal{I}_a$ is the required model of $\mathcal{T}$. First note that, as a straightforward induction shows, we have $d \in C^{\mathcal{I}_a}$ if and only if $d \in C^{\mathcal{I}_i}$ for all $d \in \Delta^{\mathcal{I}_i}$ and all $\mathcal{EL}$ concepts C (this argument fails in the case of $\mathcal{ELI}$). Consequently, the elements $d \in \Delta^{\mathcal{I}_a} \setminus \{a\}$ satisfy the TBox $\mathcal{T}$ in the sense that $d \in C^{\mathcal{I}_a}$ implies $D^{\mathcal{I}_a}$ for all $C \sqsubseteq D \in \mathcal{T}$. We

have to show that the same is true for a and that $a \in C^{\mathcal{I}_a}$ if and only if $X_C \in \Gamma_a$ for all $C \in \mathsf{sub}(\mathcal{T})$. We concentrate on the latter since, due to the rules of kind (iv), it implies the former.

The proof is by induction on the structure of C, where the induction start (C a concept name) is immediate by definition of $\mathcal{I}_a$. The case $C = D \sqcap E$ is straightforward based on the induction hypothesis, the semantics, and the rules of the form (ii). Thus consider the case $C = \exists r.D$.

(if) $X_{\exists r.D} \in \Gamma_a$ implies that $\exists r.D = \exists r_i.C_i$ for some i. By our choice of $\mathcal{I}_i$, $d_i \in C_i^{\mathcal{I}_i}$ and thus $d_i \in C_i^{\mathcal{I}_a}$. By construction of $\mathcal{I}_a$, we thus have $a \in (\exists r.C)^{\mathcal{I}_a}$.

(only if) By construction of $\mathcal{I}_a$, $a \in (\exists r.C)^{\mathcal{I}_a}$ implies $r = r_i$ and $d_i \in C^{\mathcal{I}_a}$ for some i. The latter yields $d_i \in C^{\mathcal{I}_i}$, which by our choice of $\mathcal{I}_i$ implies $\mathcal{T} \models C_i \sqsubseteq C$. From the latter, it easily follows that $\mathcal{T} \models \exists r.C_i \sqsubseteq \exists r.C$. Thus $X_{\exists r_i.C_i} \in \Gamma_a$ and the rules of kind (iv) give us $X_{\exists r.C} \in \Gamma_a$. □

We summarise the obtained result as follows.

Theorem 7.31. *For every atomic query $A_0(x)$ and $\mathcal{EL}$ TBox $\mathcal{T}$, there is a Datalog-rewriting Π of size polynomial in the size of $\mathcal{T}$.*

The material presented in this section shows that query answering with respect to TBoxes formulated in description logics such as $\mathcal{EL}$ and $\mathcal{ELI}$ is closely related to query answering in Datalog. One main difference is that DLs allow existential quantification on the right-hand side of concept inclusions, whereas Datalog does not admit existential quantification in the rule head. Inspired by this difference, researchers have generalised Datalog with this kind of existential quantification, which leads to what is known as *existential rules* or *tuple-generating dependencies*. However, query answering with respect to sets of existential rules turns out to be undecidable and, therefore, various syntactic restrictions have been proposed that regain decidability, known under the name *Datalog*$^{\pm}$. In fact, there are many different versions of *Datalog*$^{\pm}$, several of which generalise the description logics DL-Lite, $\mathcal{EL}$ and $\mathcal{ELI}$.

7.4 Complexity aspects

In this chapter, we have focussed on rewriting-based approaches to query answering in the presence of Description Logic TBoxes. As alternatives, researchers have developed approaches that materialise the consequences

of the TBox in the database instead of rewriting the query, as well as approaches that do not rely on existing database technology at all. One important motivation for the latter is to avoid the exponential blowups that are often inherent in rewriting-based approaches. Another is to enable TBox-aware querying for description logics that are too expressive to be rewritten into query languages such as SQL or Datalog. The latter is closely related to the subject of data complexity, which we briefly discuss in this section. When studying computational complexity, it is more convenient to work with decision problems than with computation problems. Therefore, from now on we will assume that queries are Boolean and speak of query entailment rather than query answering. In principle, though, everything said in the following also carries over to queries with answer variables.

Query answering in DLs is a problem with multiple inputs: the ABox (from now on called the data), the TBox and the query. The most obvious way to measure the complexity is to treat all inputs equally, which is called *combined complexity*. For example, conjunctive query entailment in the presence of TBoxes that are formulated in DL-Lite or in $\mathcal{EL}$ is NP-complete in combined complexity, and the same problem is ExpTime-complete in combined complexity when the TBox is formulated in $\mathcal{ELI}$ or in $\mathcal{ALC}$, and even 2-ExpTime-complete in $\mathcal{ALCI}$. In fact, conjunctive query entailment is NP-complete already without any TBoxes (it is then simply the homomorphism problem on directed graphs).

However, a moment of reflection reveals that combined complexity is probably *not* the most relevant form of complexity for query answering. In typical applications, the data is extremely large while the query and also the TBox are many orders of magnitude smaller. In database research, this observation has led to the notion of *data complexity*, which in the DL version reads as follows: when analyzing the complexity of query entailment, consider the ABox to be the only input while treating both the query and the TBox as parameters that are fixed and whose size therefore is a constant and does not contribute to the complexity. In effect, transitioning from combined complexity to data complexity thus means replacing a single decision problem (with three inputs) by infinitely many decision problems (with one input each): one problem for each query and each TBox.

The data complexity of a querying problem is typically much lower than its combined complexity and often reflects practical feasibility

much better. For example, SQL query entailment (without TBoxes) is PSPACE-complete and thus intractable in combined complexity, but in the extremely small complexity class AC^0 (which is below LOGSPACE and PTIME) in data complexity. Conjunctive query entailment is therefore also in AC^0 in data complexity while Datalog query entailment is known to be PTIME-complete in data complexity.

The results on rewriting presented in this chapter allow us to infer results on data complexity. First consider conjunctive query entailment in the presence of DL-Lite TBoxes. Entailment of the (fixed) conjunctive query q with respect to the (fixed) TBox $\mathcal{T}$ is reduced to entailment of their FO-rewriting $q_{\mathcal{T}}$. As noted above, the latter problem is in AC^0 in data complexity (that is, with $q_{\mathcal{T}}$ regarded as fixed). In fact, if we neglect representational differences between an ABox $\mathcal{A}$ and the corresponding interpretation/relational instance $\mathcal{I}_{\mathcal{A}}$ (which can safely be done), entailment of q with respect to $\mathcal{T}$ and entailment of $q_{\mathcal{T}}$ are *exactly the same problem*: the inputs are identical and the "yes"-inputs also coincide. Consequently, conjunctive query entailment in DL-Lite is in AC^0 in data complexity. Note that the exponential size of $q_{\mathcal{T}}$ is irrelevant since $q_{\mathcal{T}}$ is fixed and not an input. Arguing in the same way and utilising the PTIME data complexity of Datalog query entailment, we can derive from the rewritings presented in Section 7.3.2 that atomic query entailment in $\mathcal{EL}$ and in $\mathcal{ELI}$ is in PTIME regarding data complexity. In fact, it is known to be PTIME-complete, and the complexity does not change when we replace atomic queries with conjunctive queries.

Thus, query entailment in DL-Lite, $\mathcal{EL}$ and $\mathcal{ELI}$ is *tractable* in data complexity. In contrast, the use of description logics that include disjunction typically leads to intractability in data complexity. As an example, consider the following $\mathcal{ALC}$ TBox and Boolean CQ:

$$\begin{aligned} \mathcal{T} \;=\; \{ \quad & \top \sqsubseteq R \sqcup G \sqcup B \\ & R \sqcap \exists r.R \sqsubseteq D \\ & G \sqcap \exists r.G \sqsubseteq D \\ & B \sqcap \exists r.B \sqsubseteq D \quad \}, \\ q \;=\; & \exists x\, D(x). \end{aligned}$$

If we assume that the input ABox $\mathcal{A}$ contains only the role name r and no other symbols, thus representing a directed graph, then it is straightforward to prove that $(\mathcal{T}, \mathcal{A}) \models q$ if and only if $\mathcal{A}$ is not 3-colourable (counter models correspond to 3-colourings). Note that the concept name D represents a *defect* in the 3-colouring and the existence

of defects is what q queries for. Consequently, CQ entailment is coNP-hard in data complexity in $\mathcal{ALC}$ and it is not hard to lift this result to the entailment of atomic queries (which do not admit the existential quantification used in the query above). A coNP upper bound can be established in various ways, such as through tableau algorithms, resolution or construction of a model-theoretic nature. Implicit forms of disjunction (see the discussion of convexity in Section 6.3.2) also easily lead to intractability in data complexity.

7.5 Historical context and literature review

Query answering over Description Logic knowledge bases has a long tradition and can be traced back to the very beginnings of the field. Originally, the most common choice for the query language was *concept queries*, that is, queries of the form $C(x)$ with C a DL concept, typically formulated in the description logic that is also used for the TBox. Over the years, query answering has become more and more important, and the setups and questions that have been considered have become more database-like in spirit. Conjunctive queries were first considered in a DL context in [LR98] and a little later in [CDGL98a]. Together with unions of conjunctive queries (UCQs), they are now the most common query language for DLs. A very large body of literature on the topic is available; in the following, we will give references relevant for this chapter, following roughly the order in which we have presented the material.

The DL-Lite family of description logics was introduced in [CDL$^+$07], where the notions of query rewriting and FO-rewritability were also first considered in a DL context.[4] DL-Lite is the foundation for an approach to querying and integration of relational databases using ontologies and schema mappings that is called ontology-based data access (OBDA), discussed in detail in [CDL$^+$09]. Gaifman locality, as used in the proof of Theorem 7.8 to analyse the limits of FO-rewritability, can be found in most textbooks on first-order logic and finite model theory such as [Lib04]. Universal models have been used under various names in the literature on query answering with DLs, for example in [CDL$^+$07, LTW09, KZ14, BO15]. They can be viewed as a DL version of the chase procedure from database theory; see, e.g., [DNR08] and references therein. FO-rewritability of conjunctive queries in DL-Lite

[4] Query rewriting is also a popular tool in various subfields of database theory such as query answering under views [Hal01, Len02].

was first established in [CDL+07] by a procedure called PerfectRef. A more semantic approach based on so-called tree witnesses is presented in [KZ14]. Implemented systems for computing rewritings are available, such as OnTop [RKZ13]. Lower bounds on the size of rewritten queries are established in [GKK+14]. An alternative to query rewriting which materialises the consequences of the TBox in the ABox instead of anticipating them in the query was introduced in [LTW09] and applied to DL-Lite in [KLT+10]. This approach is also known as the combined approach.

Due to the limited expressive power of DL-Lite, query answering in more expressive DLs has also received significant interest. Datalog rewritings of concept queries in Horn DLs such as $\mathcal{ELI}$ were first studied in [HMS07]. Implemented systems are available, such as Rapid [TSCS15], Requiem [PUMH10] and Clipper [HS12]. A recent survey on query answering in Horn DLs is provided by [BO15]. One can also try to construct FO-rewritings of queries in Horn DLs beyond DL-Lite, although in general they are not guaranteed to exist. For concept queries, foundations are laid in [BLW13] and it is shown in [HLSW15] that FO-rewritings often exist in practice and can be computed efficiently.

For $\mathcal{ALC}$ and its extensions, even Datalog rewritings are not guaranteed to exist. One possibility is to rewrite into disjunctive Datalog instead [HMS07]; another is to give up on rewritings and implement query answering systems from scratch, based for example on tableau algorithms or resolution. Again, one can also try to construct FO-rewritings or Datalog-rewritings when they exist, as studied for example in [BtCLW14] and [GMSH13].

Both the combined complexity and the data complexity of query answering in DLs have received significant interest. Data complexity results for DLs first appeared in [DLNS98], where a coNP lower bound is established for concept queries and (a fragment of) $\mathcal{ALC}$. Corresponding coNP upper bounds can be established for conjunctive queries and a wide range of expressive DLs using various methods; see, e.g., [HMS07, OCE08, GLHS08]. Horn DLs typically have PTime data complexity, but classes such as AC^0, LogSpace and NLogSpace also play a role; see [KL07, Ros07a, CDL+13] for a sample of results. A more fine-grained approach is taken in [LW12, BtCLW14], where data complexity is studied for single TBoxes and queries, instead of for entire description logics.

Regarding combined complexity, it is a classical result in database theory that conjunctive query entailment is NP-complete [CM77]. The

combined complexity of answering both concept and conjunctive queries in $\mathcal{ALC}$ is ExpTime-complete [Lut08]. For the latter (but not for concept queries), the complexity rises to 2ExpTime-complete when inverse roles are added [Lut08]. Transitive roles and nominals are also known to increase combined complexity [ELOS09, NOS16]. In Horn DLs, the combined complexity of answering conjunctive queries is typically in ExpTime, even with inverse roles [EGOS08, ORS11]. For DLs for which subsumption is ExpTime-complete, such as $\mathcal{ELI}$, this problem (trivially) is also hard for ExpTime. Otherwise, it often turns out to be NP-complete, such as in DL-Lite (implicit in [CDL$^+$07]) and in $\mathcal{EL}$ (simultaneously observed in [KL07, KRH07, Ros07b]).

When viewed from a database perspective, one obvious shortcoming of DLs is their restriction to unary and binary relations (concept and role names). To overcome this limitation, DLs with higher arity have been proposed, e.g., in [CDL08, CDL$^+$13]. Another option is to give up DL syntax and instead use rule-based formalism as ontology languages, which naturally allow for relations of any arity. This approach has led to the development of the Datalog$^\pm$ family of ontology languages, also known as existential rules and tuple-generating dependencies. The literature on Datalog$^\pm$ has already become rather large, and we only mention [CGL12, CGK13, BLMS11, BMRT11, CGP11] to get the reader started.