QUERYING DESCRIPTION LOGIC KNOWLEDGE BASES

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2007

By
Birte Glimm
School of Computer Science

Contents

Word Count 72.357

List of Figures

Abstract

Knowledge representation systems provide a mechanism for storing facts about some part of the real world in a knowledge base, inferring new knowledge based on the given facts, and querying knowledge bases. The ability to infer new knowledge is one of the distinguishing features compared to databases. Such inference services require the definition of knowledge in a language for which such inference algorithms exist, e.g., a Description Logic (DL). A DL language allows for the specification of concepts, individuals that are instances of these concepts, and roles, which are interpreted as binary relations over the individuals.

Description Logics have proved useful in a wide range of applications and form the foundations of the Web Ontology Language (OWL), which is used in the Semantic Web as a means for specifying machine processable information.

Despite their popularity, the query facilities provided by DL systems are still limited. Current algorithms are incomplete or impose restrictions on the types of allowed queries. In this thesis we identify sources of incompleteness in existing algorithms and present extended query procedures that eliminate the deficiencies described above. More precisely, we present query answering algorithms for unrestricted conjunctive queries for the DLs $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$—the former of which was a long standing open problem. Furthermore, the correctness of the presented algorithms is proved formally and an analysis of the theoretical complexity is given. The planned future work is targeted on optimisation techniques to improve the algorithms' practicality.

The work presented in this thesis should be of value mainly to implementors of Description Logic systems, as the presented algorithms build the theoretical foundation for implementable query answering interfaces. Additionally, the algorithms can also be used in order to extend a DL system with datalog style rules.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

First of all, I would like to thank my supervisors Ian Horrocks and Uli Sattler for their continuous support, encouragement, ideas, and advice. I would also like to thank Carsten Lutz for all his input and for learning that there is no “h” in my name.

I am deeply grateful to my parents for supporting me whenever I need it. I thank Frank for believing in me, supporting me, and being there for me in many difficult days.

I would like to thank Yevgeny for caring so much about me and making me laugh even if I do not feel like it and Nestan for being as she is. It is a pleasure to have you around. I also thank everybody in the Information Management Group. In particular everybody from the coffee round. Thanks to everybody in room 2.116; I always enjoyed being around. Also thanks to Bijan: when suffering cannot be avoided, then it is better to suffer together. There are many more people, who I like to thank: my friends in Germany and many new ones I found here. Sorry for not listening everybody by name.

Finally, I would like to thank the Foundation of German Business (Stiftung der Deutschen Wirtschaft) and the Engineering and Physical Sciences Council for supporting me with a scholarship.

Chapter 1

Introduction

The aim of this research is to investigate the logical basis for improving the usability of *knowledge representation systems*, especially the improvement of the query answering facilities of such systems. A knowledge representation system (KRS) allows the storage of constraints and facts about some part of the real world in a *knowledge base* (KB). A distinguishing feature of knowledge representation systems is that they provides inference services which allow for deducing facts that are only implicitly given. With queries, users can retrieve knowledge stored in a knowledge base; either knowledge that has been directly entered into the system or inferred knowledge. Currently available reasoning systems, such as FaCT++ [133], KAON2 [92], RACER [57], Pellet [123], SPASS [142], and MSPASS [72], already offer very expressive languages for the specification of knowledge and the inference services of these systems are very powerful and highly optimised. The query languages for these systems are, however, still limited. Few provably correct algorithms for expressive query languages are known, and, furthermore, we show that some of the proposed query answering algorithms [22, 98] for very expressive Description Logics are incomplete, i.e., they do not necessarily return all answers of a query.

The aim of this thesis is, therefore, to devise extended query algorithms for expressive Description Logics, which can serve as a basis for implementation and optimisation in existing DL reasoners.

This chapter informally introduces the underlying foundations of DL based knowledge representation systems; this includes the languages used to specify knowledge, available inference services, common application areas of knowledge representation systems, and ways of querying knowledge bases. Based on these

foundations, the problems of query answering that remain to be solved are highlighted and the aims and objectives of this thesis are stated.

1.1 Description Logics

Description Logics [8] are knowledge representation languages with a formally defined syntax and semantics. A Description Logic allows for the specification of concepts (also known as classes), individuals (also known as objects) that are instances of these concepts, and roles (also known as properties) that are interpreted as pairs of individuals that are related by the role. Operators, such as negation ($\neg$) or conjunction ($\sqcap$), can be used in order to build more complicated composite concepts. As an example, consider the following concept:

$$Female \sqcap \geqslant 2\ hasChild \sqcap \forall hasChild.Female$$

In natural language, this concept describes females who have at least two children and all of whose children are female. The description consists of three subconcepts connected with the conjunction operator ($\sqcap$). The first one describes objects that belong to the atomic concept *Female*, the second one describes objects with at least two objects related via the role *hasChild*, and the third concept covers all objects for which *all* ($\forall$) objects related via the role *hasChild* are instances of the concept *Female*.

A particular Description Logic is characterised by its set of supported constructors. Schmidt-Schauss and Smolka [119] introduced a naming scheme for Description Logics and called the basic language $\mathcal{AL}$ (Attributive concept Language). The logic $\mathcal{AL}$ allows for negation of atomic concepts (e.g., $\neg Female$), conjunction ($\sqcap$), universal quantification ($\forall hasChild.Female$), and unqualified existential quantification ($\exists hasChild$), which means that all instances of this concept have a *hasChild*-successor. For additional operators a further letter is appended, e.g., $\mathcal{AL}$ plus *complex negation* is called $\mathcal{ALC}$. Complex negation allows the negation of a complex concept expressions as $\neg(\forall hasChild.(Female \sqcap Doctor))$ and not only the negation of an atomic concept as in $\neg Female$, where no further definition of $Female$ is given. A more detailed introduction to Description Logics follows in Chapter 2.

1.1.1 Description Logic Knowledge Bases

Using a Description Logic language, users can build a *terminology* of agreed terms and use a knowledge representation system to store and reason about such a terminology. A terminology, also called a $TBox$, is a set of axioms that induce a concept hierarchy, e.g., the axiom $Student \sqsubseteq Person$ states that the concept $Student$ is a specialisation of (or, is subsumed by) the concept $Person$ and therefore every instance of the concept $Student$ must also be an instance of the concept $Person$.

Description Logics may also allow for a *role hierarchy*, also called an $RBox$, where, for example, the axiom $hasSon \sqsubseteq hasChild$ states that all pairs of individuals that are related via the role $hasSon$ are also related via the role $hasChild$. If a DL in addition supports transitive roles, one can, for example, state that the role $hasDescendant$ is transitive. A DL reasoner can then deduce that, if $hasDescendant(Mary, Peter)$ and $hasDescendant(Peter, Bill)$ holds, then $hasDescendant(Mary, Bill)$ also holds.

Assertions about individuals have the form $Female(Mary)$, $hasChild(Mary, Peter)$, or $Mary \neq Maria$, and are called concept assertions, role assertions, and inequality assertions, respectively. The first assertion states that the individual $Mary$ is an instance of the concept $Female$ (in natural language: Mary is a female), the second one states that the individual $Mary$ is related to the individual $Peter$ with the role $hasChild$ or, in other words, $Peter$ is a $hasChild$-successor of $Mary$ and $Mary$ is a $hasChild$-predecessor of $Peter$ (in natural language: Peter is a child of Mary). The third assertion simply states that $Mary$ is a different individual from $Maria$. A collection of assertions about individuals is called an $ABox$.

A TBox, RBox (if supported by the language), and ABox together constitute a *knowledge base* (also known as *ontology*).

In this thesis, we consider different expressive DLs and all of them allow for role hierarchies, transitive roles, and qualified number restrictions of the form $\geqslant 2\, hasChild.Female$. The latter concept describes individuals who have two or more $hasChild$-successors that are females. The DL $\mathcal{SHIQ}$, which we use in Chapter 4, additionally allows for inverse roles. With inverse roles we can, for a role $hasChild$, use $hasChild^-$ as the role that, intuitively, represents the $isChildOf$ relation. In Chapter 5, we present an algorithm for the DL $\mathcal{SHOQ}$ that supports nominals. With nominals, we can define a concept by enumerating

its instances, e.g., the concept *EU* can be defined as an enumeration of its member countries.

Different DLs may have different computational properties. The time needed to decide if, for example, one concept is more general than another (called a *subsumption check*), is polynomial in the size of the input concepts for very inexpressive DLs, such as $\mathcal{FL}^-$ [18] or $\mathcal{EL}$ [21]. For very expressive DLs, the complexity of this problem increases, e.g., it is ExpTime-complete for $\mathcal{SHIQ}$ [132] and even NExpTime-complete for $\mathcal{SHOIQ}$ [99, 131], which allows for both: nominals and inverse roles. If the knowledge to be modelled does not require high expressivity, applications can benefit from systems such as CEL [9]. CEL supports the DL $\mathcal{EL}^+$ and implements an algorithm that runs in a polynomial time for the standard reasoning problems.

1.1.2 Historical Background of Description Logics

Early knowledge representation systems include semantic networks [85, 103] and frames [90]. Semantic networks use directed graphs, in which nodes represent concepts and edges—or so called arcs—represent relations between concepts. Despite being called semantic networks, a major shortcoming of these early systems is that the semantics of the relations is not formally defined. Arcs can represent various kinds of relations [17, 146], e.g., a part-of relation or a sub-concept relationship (sometimes called sub-type, specialisation or IsA-relation). A frame represents a concept, it has an identifier (ID), and it can have slots or fillers, i.e., elements from a concrete domain (e.g., integers) or IDs of other frames. These slots describe attributes of the objects, together with default values, procedures to compute values or to propagate side effects, and restrictions on possible fillers. Attributes in these systems correspond to roles in modern DLs.

Brachman and Schmolze [19] first tried to provide a well-defined semantics for semantic networks and frames with their system Kl-One. This system was succeeded by systems based on Description Logics (DL), e.g., Loom [89], Krypton [20], Nikl [76], Back [102], Kris [7], Classic [16], and others.

Inference algorithms for these systems were often based on analysis and/or decomposition of the syntax or structure of the knowledge base statements and were, therefore, called structural comparison algorithms [36]. Some of these systems already provided very sophisticated query facilities, e.g., Loom, but the

algorithms of these systems typically suffered from incompleteness or were restricted to very simple languages. An algorithm that decides, given two concepts C and D, whether it holds that D subsumes C, is called complete if it is guaranteed that the algorithm returns "yes" whenever the subsumption relationship holds and it is called incomplete otherwise. The algorithm is called sound if it is guaranteed that, whenever the algorithm returns "yes" that D indeed subsumes C and it is called unsound otherwise. Finally we call such an algorithm *correct*, if it is both sound and complete. An algorithm that is guaranteed to return an answer after finitely many steps is called *terminating*. A *decision procedure* for a problem, e.g., subsumption, is an algorithm that is sound, complete, and terminating. Please note that a decision procedure is formulated for a particular decision problem (here subsumption between two concepts), i.e., a problem that has a Boolean answer ("yes/no" or true/false).

Many current systems for expressive Description Logics, e.g., FaCT++ [133], RACER [57], and Pellet [123], are based on tableau algorithms and employ highly optimised decision procedures for subsumption checking, consistency checking (i.e., checking whether a knowledge base is contradictory and cannot have a model), or instance retrieval (i.e., for retrieving all individual names that belong to a given, possibly complex, concept). Resolution-based decision procedures are, however, also used, for example, in the First-Order Logic reasoners SPASS [142] and MSPASS [72] that implement decision procedures for expressive Description Logics and, more recently, in the DL reasoner KAON2 [92]. In Section 1.2, we give more more information about standard reasoning services and we introduce queries over DL knowledge bases.

1.1.3 Application Areas of Description Logics

Knowledge representation systems have been used in several practical applications. A short list of examples is given below, a much broader description of application areas is given in the Description Logic Handbook [8].

Semantic Web The Semantic Web [11, 12] aims to make the information on the World Wide Web processable for computer programs (agents) by annotating resources on the web with terms whose meaning is defined in an ontology. As a language to describe the terms in an ontology, the World Wide Web Consortium (W3C) standardised the Web Ontology Language OWL [10]. OWL has three

sub-species, called OWL Lite, OWL DL, and OWL Full.[1] OWL Lite corresponds to the DL $\mathcal{SHIF}(\mathbf{D})$ ($\mathcal{SHI}$ plus functional restrictions and a restricted form of concrete domains) and OWL DL corresponds to $\mathcal{SHOIN}(\mathbf{D})$ ($\mathcal{SHI}$ plus nominals, unqualified number restrictions and a restricted form of concrete domains). OWL Full does not correspond to a Description Logic, and the standard reasoning problems are undecidable. Recently, OWL 1.1 [94] was proposed, which extends OWL DL with useful and yet practical features (i.e., implemented systems are already available [123, 133]). In September 2007 the W3C approved the OWL Working Group[2] that is to produce a W3C Recommendation that refines and extends OWL in this direction.

Medical Informatics In particular in this area, several very large ontologies are available. For example, GALEN[3] provides a clinical terminology with nearly 25,000 atomic concepts to support the management of clinical information [104, 147]. The SNOMED ontology [124, 125, 126] provides a comprehensive terminology for health care with more than 250,000 atomic concepts, but a rather simple structure. The National Cancer Institute[4] (NCI) provides a terminology with more than 25,000 concepts that is used in clinical care, translational and basic research, and for public information and administrative activities [33]. Other examples in this area include ontologies about anatomy [50, 120].

Life Sciences The myGrid project uses knowledge representation to organise workflows for experiments relying on web-based procedures and to describe how and why results were produced [37, 39, 48, 70, 134, 148]. Several ontology based methods are also used by biologists, e.g., to associate genotype and phenotype information [112] or to classify proteins [144]. Well known ontologies from this area include the Gene Ontology (GO) or the Open Biomedical Ontologies (OBO) [49, 51, 96].

Other application areas include software engineering [58], configuration [110], information integration [23], biology [121, 145, 148], geography [52], geology [127], and defence [84].

[1] Inference in OWL Full is clearly undecidable as OWL Full does not include restrictions on the use of transitive properties, which are required in order to maintain decidability [66].

[2] `http://www.w3.org/2007/06/OWLCharter`

[3] `http://www.opengalen.org`

[4] `http://www.nci.nih.gov/`

1.1.4 Semantics of Description Logics

Description Logics have a well defined semantics, which means that compared to the older frame systems or semantic networks there is no ambiguity between subsumption relationships, role relationships or individuals and concepts in general. The semantics of the language constructors and inference tasks is also precisely defined. The semantics is given by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a non-empty set of individuals $\Delta^{\mathcal{I}}$ called the interpretation domain, and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps atomic concepts to subsets of the domain $\Delta^{\mathcal{I}}$, atomic roles to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and individual names to elements of the set $\Delta^{\mathcal{I}}$. From this, one can also inductively interpret the complex concepts, e.g., a conjunction of two atomic concepts C and D is interpreted as the intersection of the two sets $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$. For a concept C, we call the set $C^{\mathcal{I}}$ the extension of C. We introduce interpretations more precisely in the following chapter.

A concept C is called *satisfiable* if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}}$ is non-empty and it is *unsatisfiable* otherwise. The concept $Female \sqcap \neg Female$, for example, is unsatisfiable independent of all other facts that might be specified in the knowledge base since in every interpretation $\mathcal{I}$, $\neg Female$ is interpreted as the set of elements of the domain that are not in the set $Female^{\mathcal{I}}$, i.e., elements that are in $\Delta^{\mathcal{I}} \setminus Female^{\mathcal{I}}$. We can abbreviate such an unsatisfiable concept with $\bot$ and denote it as *bottom*. On the contrary, the concept $Female \sqcup \neg Female$ is always satisfiable and we call such a concept the *top* concept and abbreviate it with $\top$.

Obviously there can be many interpretations for a knowledge base. If the ABox contains, for example, an assertion $(Doctor \sqcup Lawyer)(Andrea)$ (i.e., Andrea is a doctor or a lawyer), then there could be interpretations in which $Andrea^{\mathcal{I}} \in Doctor^{\mathcal{I}}$ and others in which $Andrea^{\mathcal{I}} \in Lawyer^{\mathcal{I}}$ or both. Interpretations that, informally, respect all facts given in a knowledge base, i.e., if $Female(Mary)$ is present in the ABox, then $Mary^{\mathcal{I}} \in Female^{\mathcal{I}}$, are called *models* for the knowledge base.

1.2 Reasoning Services

Besides just storing the given facts about a domain in a knowledge base, users want to interact with the system. For example, they want to determine whether

the given facts are contradictory. Current reasoners offer, therefore, a more or less expressive query language.

1.2.1 Standard Reasoning Services

All reasoners use algorithms for inferring implicitly stated knowledge and, based on these algorithms, they also support some basic reasoning tasks, e.g., determining if a given individual is an instance of a given concept. Systems usually support the following standard reasoning tasks:

1. Subsumption Test: to determine whether one concept is more general than another. Subsumption tests are used to build and maintain a taxonomy of named concepts, called the concept hierarchy, and the process of building the concept hierarchy is called classification.

2. Satisfiability Test: to determine whether the constraints implied by the knowledge base are such that a concept is contradictory and thus its extension is empty in every model of the knowledge base.

3. Consistency Test: to determine whether a given knowledge base can have a model.

4. Retrieval: to retrieve all instances of a given concept or all pairs of individuals related via a given role.

However, users often want to use more complex queries. For example, a query that asks for a triple consisting of a person plus their father and their mother cannot simply be reduced to a retrieval query for concept or role instances. Conjunctive queries, which are similar to SQL queries in a database system, provide a more expressive query language and we briefly introduce them in the following section.

1.2.2 Conjunctive Queries

In data-intensive applications, querying KBs plays a central role. Instance retrieval is, in some aspects, a rather weak form of querying: concepts are used as queries and concepts cannot express arbitrary relational structures. Additionally, it is not directly possible to retrieve tuples of individuals that fulfill certain

conditions. Conjunctive queries (CQs), well known in the database community, provide an expressive query language with capabilities that go beyond standard instance retrieval.

A conjunctive query is a conjunction of concept expressions of the form $C(t)$ and role expressions of the form $r(t, t')$, where C is a concept, r is a role, and t, t' are terms, i.e., variables or individual names. If $Father$ and $Mother$ are concepts and $hasChild$ is a role in the queried knowledge base, a query to retrieve a triple of father, mother and their child could be expressed as:

$$(x, y, z) \leftarrow Father(x) \wedge Mother(y) \wedge hasChild(x, z) \wedge hasChild(y, z)$$

The left hand side of the arrow (x, y, z) defines which variables are *distinguished* or *answer* variables. If all variables are distinguished, as in this example, the query answer consists of triples of individual names from the knowledge base such that, after replacing the variables with the corresponding individual names from the query answer, all conjuncts are true in each model of the knowledge base. For example, let the knowledge base to be queried consist of the following TBox and ABox:

$$\begin{aligned}
\mathcal{T} = \ & \{ \\
& Male \sqcap \exists hasChild.\top \sqsubseteq Father && (1.1) \\
& Female \sqcap \exists hasChild.\top \sqsubseteq Mother && (1.2) \\
& \} \\
\mathcal{A} = \ & \{ \\
& hasChild(Peter, Paul) && (1.3) \\
& hasChild(Mary, Paul) && (1.4) \\
& Male(Peter) && (1.5) \\
& Female(Mary) && (1.6) \\
& \}
\end{aligned}$$

The triple $(Peter, Mary, Paul)$ is an answer for the query since $Father(Peter)$ is a consequence of 1.5, 1.3, and 1.1, $Mother(Mary)$ holds due to 1.4, 1.6, and 1.2, and the last two conjuncts $hasChild(Peter, Paul)$ and $hasChild(Mary, Paul)$ are a directly stated in the ABox by the assertions 1.3 and 1.4 respectively.

Query answering with respect to a knowledge base is different from query answering with respect to a database. In databases we have the *Closed World Assumption* (CWA), which means that everything that is not explicitly stated is assumed to be false. In DL systems and in First-Order Logic (FOL) in general, we usually have an *Open World Assumption* (OWA) and, therefore, only incomplete information about the domain. Consider, for example, a knowledge base that contains only the two ABox assertions $supervises(Ian, Birte)$ and $Female(Birte)$. In a DL system, the query

$$(x) \leftarrow (\forall supervises.Female)(x),$$

has no answer since there can be a model of the knowledge base in which Ian also supervises a male student. In a database system, the closed world assumption means that we have complete knowledge about the domain, i.e., Ian does not supervise any student unless explicitly stated. Hence, a database would return Ian as an answer to the above query.

In database systems, we also have the *Unique Name Assumption* (UNA), which means that different individual names are mapped to distinct elements in the model. In current DL systems we usually do not employ the UNA, which means that different individual names can be mapped to the same element in a model.

As a consequence of the fact that DL systems have the OWA (and do usually not make the UNA), a DL knowledge base has usually more than one model, whereas a database always has only a single model. Query answering in a database setting requires, therefore, just checking one model, whereas in our setting it requires checking all models of a knowledge base, i.e., checking logical entailment. This explains why query answering with respect to a DL knowledge base is a (computationally) harder task than query answering with respect to a database.

If the query also contains non-distinguished variables, these variables are treated as existentially quantified, i.e., we just require the existence of a suitable element in the model, but this element does not have to correspond to a named individual in the ABox. Assume, for example, that the TBox of a knowledge base $\mathcal{K}$ contains the axiom

$$Father \sqsubseteq Male \sqcap \exists hasChild.Person$$

and the ABox contains only the assertion $Father(Charles)$. Then $Charles$ is a correct answer for the query

$$(x) \leftarrow hasChild(x, y),$$

where y is a non-distinguished variable. This is because all instances of the concept $Father$, and therefore also $Charles$, are males and they have a $hasChild$-successor that is an instance of the concept $Person$. From this, it follows that $Charles$ has a $hasChild$-successor, although the name is not known.

In order to find the answers for this particular query, one can use the standard instance retrieval techniques since finding the answers for this query is equivalent to retrieving all instances of the concept $\exists hasChild.\top$. The reduction of a query to a concept by eliminating variables is often called the rolling-up [129] or tuple graph [22] technique. The term rolling-up stems from the systematics behind this transformation, and the technique is easier to explain when the query is represented as a graph, called a query or tuple graph. Understanding the meaning of longer queries can also be easier with the graph representation.

Example 1.1.

$$\begin{aligned}(u) \leftarrow \quad & hasFriend(u, x) \wedge hasChild(x, y) \wedge hasChild(x, y') \wedge \\ & Doctor(y) \wedge Lawyer(y')\end{aligned}$$

The query from Example 1.1 is represented by the graph in Figure 1.1. Informally, each variable v in the query gives rise to a node in the graph, labelled with a conjunction of concepts such that, for each conjunct C, $C(v)$ is a concept conjunct in the query. Each role conjunct $r(v, v')$ gives rise to an edge in the graph going from the node that represents v to the node that represents v' and labelled with r.

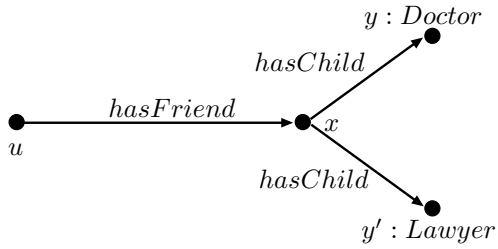

Figure 1.1: A graphical representation of a query.

In order to reduce the query from Example 1.1 to a concept, we traverse the query graph in a depth-first manner, starting at the node that represents the distinguished variable. When reaching a leaf node v, we remove the node and its incoming edge (v', v), and conjoin the concept $\exists r.C$ to the label of v', where r is the label of the replaced edge and C is the label of v.

We can, for instance, remove the node that represents y and its incoming *hasChild* edge by setting the label of the node for x to $\exists hasChild.Doctor$. This concept still captures the required existence of some *hasChild*-successor who is a *Doctor*. We can proceed, similarly, for the node that represents y' and its incoming edge, resulting in the label

$$\exists hasChild.Doctor \sqcap \exists hasChild.Lawyer$$

for the node that represents x. Finally, the query graph can be collapsed into a single node labelled with

$$\exists hasFriend.(\exists hasChild.Doctor \sqcap \exists hasChild.Lawyer).$$

We can now compute the query answers by retrieving all instances of the above concept.

For simplicity, we use the variable names occurring in the query directly as nodes in the query graph. More formally, one should define a different set of nodes and a one-to-one mapping between the query and its graph representation. Furthermore, it is not hard to see that, in the absence of inverse roles, this technique cannot always be applied as straightforwardly as shown above. If, for example, the edge from x to y went from y to x instead, one can no longer replace the node y and the edge (y, x) as one would need the concept expression $\exists hasChild^{-}.Doctor$. Tessaris [129] shows, however, how this technique can be extended to the logic $\mathcal{SHF}$ (that is $\mathcal{ALC}$ plus transitive roles, role hierarchies, and functional roles, but without inverses).

Another problematic situation arises when the query contains two different role conjuncts for the same pair of terms. For example, the query $(x) \leftarrow loves(x, y) \wedge hasChild(x, y)$ asks for all individuals that love their child. If we want to express this query as a concept, we need conjunction on roles in order to capture that both role relations (*loves* and *hasChild*) must hold. If the logic allows for role conjunction, we can use the concept $\exists(loves \sqcap hasChild).\top$ to

retrieve all answers to the query. Although we are concerned with expressive Description Logics, most of the decision procedures for these logics do not support role conjunction. We present, therefore, suitably extended decision procedures in the following chapters where necessary.

Queries that do not contain distinguished variables are called *Boolean queries* and the answer to such a query is either true or false. Deciding whether a Boolean query is true or false is known as the *query entailment* problem, whereas the problem of finding all answer tuples for a non-Boolean query is known as the *query answering* problem. Although not efficient in practice, query answering can easily be reduced to query entailment by replacing the answer variables with each possible combination of individual names and by checking entailment of the resulting Boolean queries. The answers to the query are all those tuples of individual names for which the resulting Boolean query is entailed by the knowledge base. Since query entailment is the basic decision problem that also underlies query answering, we concentrate in the remainder of this thesis on the problem of query entailment. Query entailment is also well suited for analysing the theoretical complexity of an algorithm because it is a decision problem, whereas query answering is a computation problem.

1.2.3 Challenges of Query Answering

Finding a decision procedure for conjunctive query entailment for very expressive DLs such as $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOIQ}$ is not as straightforward as one might think. Even for simpler logics, there are some pitfalls that have to be avoided. The main challenge is the correct handling of the non-distinguished variables, in particular when these are arranged in a cycle. Consider, for example, the query from Example 1.2, which extends the query given in Example 1.1. The query graph for this query is depicted in Figure 1.2. We call a query cyclic when the undirected graph that is underlying its query graph is cyclic (and this is clearly the case for the given example). All other queries are called acyclic.

Example 1.2.

$$(u) \leftarrow hasFriend(u,x) \wedge hasChild(x,y) \wedge hasChild(x,y') \wedge hasChild(z,y) \wedge hasChild(z,y') \wedge Doctor(y) \wedge Lawyer(y').$$

Due to the so called tree model property of most DLs [140], a concept in most

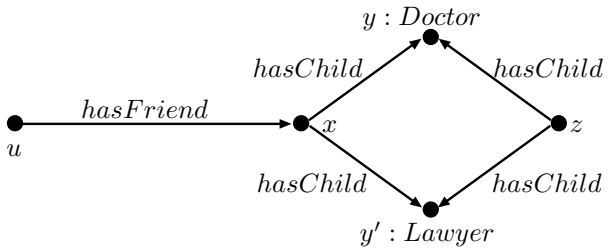

Figure 1.2: A graphical representation of a query.

DLs can only express tree-like structures. In $\mathcal{ALC}$ (and even more expressive logics), for example, we cannot express a concept $Narcist$ as those persons who love themselves. One could formulate an axiom such as $Narcist \sqsubseteq Person \sqcap \exists loves.Narcist$, but that would only guarantee that each instance d of the concept $Narcist$ has a $loves$-successor that is also an instance of the concept $Narcist$, but not that this successor is again d. It is, however, easy to define a query such as

$$(x) \leftarrow Person(x) \wedge loves(x, x),$$

where the answer would consist of all those individual names a for which a is a person who loves him- or herself. In this case, we cannot apply the rolling-up technique described above since a concept cannot express such cyclic structures. Since we can define cyclic structures in the ABox, e.g., we can state that Peter loves himself by adding the assertion $loves(Peter, Peter)$, we could argue that we can simply replace variables in cycles with individual names. We could then use the standard reasoning tasks in order to verify that the chosen individual is indeed an instance of the concept $Person$ and related to itself.

This argument does, however, not always hold. Consider, for example, the query from Example 1.2 with respect to a knowledge base containing only the assertion

$$(\exists hasFriend.(\exists hasChild.(Doctor \sqcap Lawyer \sqcap \exists hasChild.\top)))(Mary).$$

In every model $\mathcal{I}$ of the knowledge base, the individual $Mary^{\mathcal{I}}$ has a $hasFriend$-successor, say d, d has a $hasChild$-successor, say d', such that $d' \in Doctor^{\mathcal{I}}, d' \in Lawyer^{\mathcal{I}}$, and d' has again a $hasChild$-successor, say d''. It is not hard to see that d meets all the constraints for x, d' those for y and y', and d'' those for z. Since the same element in a model, here d', can be used for different variables, here

y and y', we can not just replace the variables in a cycle with individual names from the knowledge base; we also have to apply the rolling-up technique to those acyclic queries that we can obtain by identifying variables. E.g., by identifying y' and y in the query from Example 1.2, we would additionally obtain the acyclic query

$$\begin{aligned}(u) \leftarrow \quad & hasFriend(u,x) \wedge hasChild(x,y) \wedge hasChild(y,z) \wedge \\ & Doctor(y) \wedge Lawyer(y),\end{aligned}$$

depicted in Figure 1.3.

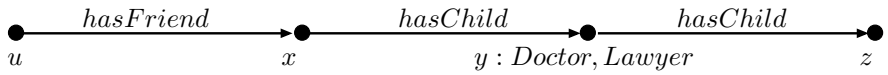

Figure 1.3: The query graph for the query obtained from Example 1.2 by identifying y' with y.

If we then reduce the query to a concept by applying the rolling-up technique, using x as the root variable, we obtain the concept

$$\exists hasFriend.(\exists hasChild.(Doctor \sqcap Lawyer \sqcap \exists hasChild.\top)).$$

Instances of this concept are answers to the query, and we would clearly retrieve $Mary$ for the given knowledge base.

The need to test all possible ways of identifying variables was first pointed out by Horrocks et al. [65]. This also applies to the tuple graph technique [22], although it is not mentioned there.

Further difficulties arise when the logic allows for transitive roles. Consider, for example, the query from Example 1.3 and its query graph in Figure 1.4.

Example 1.3.

$$\begin{aligned}(u) \leftarrow \quad & hasFriend(u,x) \wedge hasChild(x,y) \wedge hasDescendant(y,z) \wedge \\ & hasDescendant(x,z)\end{aligned}$$

We assume that the knowledge base contains, as before, the assertion

$$(\exists hasFriend.(\exists hasChild.(Doctor \sqcap Lawyer \sqcap \exists hasChild.\top)))(Mary).$$

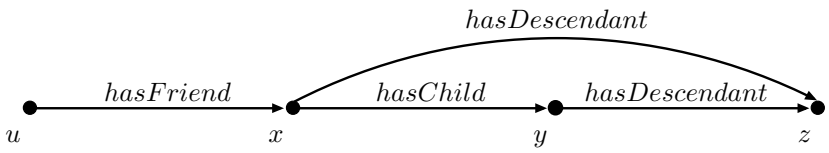

Figure 1.4: The query graph for the query from Example 1.3.

plus a role inclusion axiom that states that *hasChild* is a sub-role of the transitive role *hasDescendant*. In this case, every *hasChild*-successor is also a *hasDescendant*-successor and, additionally, *hasDescendant* is mapped to a transitive relation in each model of the KB. It is, therefore, clear that the query is not really cyclic since the edge from x to z is just a "shortcut" that is implied in any model of the KB. If we delete the conjunct $hasDescendant(x, z)$ from the query and apply the rolling-up technique, we obtain the concept

$$\exists hasFriend.(\exists hasChild.(\exists hasDescendant.\top)),$$

and we can obtain all query answers by retrieving the instances of this concept. As before, replacing the variables in the cycle with individual names from the KB would wrongly result in an empty query answer. The basic idea of modifying a given query into a simpler one that can still be used to retrieve the correct query answers is one of the basic ideas behind the algorithms that we introduce in Chapter 4 and Chapter 5 in more detail.

Tessaris [129] also uses the technique of expressing queries as concepts, but imposes two restrictions on the structure of the queries that his algorithm can handle:

1. Cycles in the query can contain only roles that are not transitive and that do not have a transitive sub-role.

2. If there are more than two role conjuncts for the same pair of variables, e.g., the query contains the conjunct $r(x, y)$ and $s(x, y)$, then one role must be a sub-role of the other or both roles must have a common functional super-role.

Tessaris formulated possible extensions of his algorithm for lifting the restrictions, but a proof that the suggested extension indeed results in a decision procedure was left for future work.

1.3 Aims and Objectives

As we have shown above, conjunctive queries allow for a flexible interaction with knowledge representation systems. Several difficulties arise, however, when devising a decision procedure for conjunctive query entailment in expressive DLs. As a result, known decision procedures are either limited to less expressive logics such as $\mathcal{ALC}$, or make restrictions on the structure of the queries, e.g., queries must be acyclic, no transitive roles are allowed in the query, or non-distinguished variables are not supported in general. The aim of this thesis is, therefore, to investigate whether it is possible to develop decision procedures for (unrestricted) conjunctive queries in expressive DLs.

1.4 A Guide for Readers

In this section we briefly summarise the chapters of this thesis and state the contributions made. References to the corresponding publications are given where relevant.

This introductory chapter is meant to be more intuitive than precise and most terms are not formally defined. In Chapter 2 we provide definitions for the Description Logics we use and introduce the important terms and concepts more precisely. Experts in Description Logics might want to skip this chapter and start directly with Chapter 3, which introduces related work, points out mistakes in existing techniques [42], and analyses the problems which arise when we try a straightforward extension of existing techniques to more expressive logics [44]. Additionally, we illustrate how techniques known from Hybrid Logics such as the downarrow binder for labelling states (elements of the domain in DL terms) can be used for conjunctive query entailment in $\mathcal{SHQ}$ [44, 69]. In Chapter 4, we introduce a decision procedure for unions of conjunctive queries in $\mathcal{SHIQ}$ [45, 47]. We further establish two tight complexity bounds: regarding combined complexity, we prove that there is a deterministic algorithm for query entailment that needs time single exponential in the size of the knowledge base and double exponential in the size of the query. We close, therefore, the long-standing open question of whether conjunctive queries are decidable for OWL Lite. The algorithm rewrites a given query into a set of structurally simpler queries that are then used to determine the query answer. In Chapter 5, we show how this technique can be

adapted to $\mathcal{SHOQ}$ [44, 46] and we establish a 2ExpTime upper bound for the combined complexity of the problem. To the best of our knowledge, it is the first query entailment algorithm for expressive Description Logics (i.e., DLs with ExpTime-complete standard reasoning tasks) that allow for nominals. Finally, the achievements and the significance of the results obtained in this thesis are outlined in Chapter 6 and remaining open questions and directions for future research are discussed.

Chapter 2

Foundations of Description Logics

To avoid ambiguities, a definition of the syntax, semantics and reasoning problems of a knowledge representation language is necessary. It was one of the shortcomings of the early frame systems that their semantics was not clear; as a consequence, the represented knowledge was interpreted differently by different people or applications. Furthermore, a proof of the correctness and completeness of an algorithm is only possible with a well defined semantics.

Description Logics have a well defined semantics, which is introduced in this chapter. Informal definitions for most of the terms are given in the introduction; the aim for this chapter is a precise definition of the terms and notation used within this thesis. We focus on the logic $\mathcal{SHOIQ}$ here, as simpler logics can easily be obtained by disallowing some of its constructors.

2.1 Syntax and Semantics

As already stated in the introduction, Description Logics allow for defining concepts, individuals that are instances of these concepts and roles that are interpreted as binary relations. In this thesis, as a convention, concept names are written in upper case, while role and individual names are written in lower case. Unless stated otherwise, we use A and B for concept names; C, D, and E for possibly complex concepts; r, s, and t for role names; a, b, and c for individual names that are used only in the ABox, and o for nominals that are used in TBox axioms or that occur in complex concepts. An integer index might be appended

if necessary.

We first define the syntax and semantics for roles, and then go on to concepts, individuals, and knowledge bases.

Definition 2.1. Let N_C, N_R, and N_I be countable, infinite, and pairwise disjoint sets of *concept names*, *role names*, and *individual names*, respectively. We assume that the set of role names contains a subset $N_{tR} \subseteq N_R$ of *transitive role names*. We call $\mathcal{S} = (N_C, N_R, N_I)$ a *signature*. The set $\mathsf{rol}(\mathcal{S})$ of $\mathcal{SHOIQ}$*-roles* over $\mathcal{S}$ (or roles for short) is $N_R \cup \{r^- \mid r \in N_R\}$, where roles of the form r^- are called *inverse roles*. A *role inclusion axiom* (RIA) is of the form $r \sqsubseteq s$ with r, s roles. A *role hierarchy* $\mathcal{R}$ is a finite set of role inclusion axioms.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$, which maps every concept name $A \in N_C$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, every role name $r \in N_R$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, every role name $r \in N_{tR}$ to a transitive binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and every individual name $a \in N_I$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. For each role name $r \in N_R$, the interpretation of its inverse role $r^{-\mathcal{I}}$ consists of all pairs $(d, d') \in \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ for which $(d', d) \in r^\mathcal{I}$. An interpretation $\mathcal{I}$ *satisfies* a role inclusion $r \sqsubseteq s$ if $r^\mathcal{I} \subseteq s^\mathcal{I}$ and a role hierarchy $\mathcal{R}$ if it satisfies all role inclusions in $\mathcal{R}$. △

For simplicity, we use the following standard notations:

1. It is clear from the semantics that the inverse relation is symmetric, i.e., the inverse of r^- is again r. To avoid writing role expressions such as r^{--} or r^{---} we define a function Inv which returns the inverse of a role. More precisely, $\mathsf{Inv}(r) = r^-$ if $r \in N_R$ and $\mathsf{Inv}(r) = s$ if $r = s^-$ for a role name s.

2. Since an inclusion relation between two roles transfers to their inverses and since set inclusion is transitive, we define, for a role hierarchy $\mathcal{R}$, $\sqsubseteq^*_\mathcal{R}$ as the reflexive transitive closure of $\sqsubseteq$ over $\mathcal{R} \cup \{\mathsf{Inv}(r) \sqsubseteq \mathsf{Inv}(s) \mid r \sqsubseteq s \in \mathcal{R}\}$. We use $r \equiv_\mathcal{R} s$ as an abbreviation for $r \sqsubseteq^*_\mathcal{R} s$ and $s \sqsubseteq^*_\mathcal{R} r$.

3. For a role hierarchy $\mathcal{R}$, we define the set $\mathsf{Trans}_\mathcal{R}$ of transitive roles as $\{r \mid$ there is a role s such that $r \equiv_\mathcal{R} s$ and $s \in N_{tR}$ or $\mathsf{Inv}(s) \in N_{tR}\}$.

4. A role r is called *simple* w.r.t. a role hierarchy $\mathcal{R}$ if, for each role s such that $s \sqsubseteq^*_\mathcal{R} r, s \notin \mathsf{Trans}_\mathcal{R}$.

The subscript $\mathcal{R}$ of $\sqsubseteq^*_{\mathcal{R}}$ and $\mathsf{Trans}_{\mathcal{R}}$ is dropped if clear from the context.

If a DL supports inverse roles, indicated by the letter $\mathcal{I}$ in the name of the DL, all role relations are bidirectional. The ABox assertion $hasChild(Mary, Peter)$ can then equally be expressed as $\mathsf{Inv}(hasChild)(Peter, Mary)$.

Definition 2.2. Given a signature $\mathcal{S} = (N_C, N_R, N_I)$, the set $\mathsf{con}(\mathcal{S})$ of $\mathcal{SHOIQ}$-*concepts* (or concepts for short) over $\mathcal{S}$ is the smallest set built inductively over symbols from $\mathcal{S}$ using the following grammar, where $o \in N_I, A \in N_C, n \in \mathbb{N}_0$, r is a role and s is a simple role:

$$\begin{aligned} C ::= \ & \top \mid \bot \mid \{o\} \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \\ & \forall r.C \mid \exists r.C \mid \leqslant n\ s.C \mid \geqslant n\ s.C. \end{aligned}$$

△

The restriction to simple roles in number restrictions is necessary to maintain decidability [66], although recent results show that, in some cases, this restriction can be lifted [79].

Definition 2.3. The semantics of $\mathcal{SHOIQ}$-concepts over a signature $\mathcal{S}$ is defined as follows:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ \bot^{\mathcal{I}} &= \emptyset \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \qquad (\{o\})^{\mathcal{I}} = \{o^{\mathcal{I}}\} \\ (\forall r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{ if } (d, d') \in r^{\mathcal{I}}, \text{ then } d' \in C^{\mathcal{I}}\} \\ (\exists r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{ There is a } (d, d') \in r^{\mathcal{I}} \text{ with } d' \in C^{\mathcal{I}}\} \\ (\leqslant n\ s.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \sharp(s^{\mathcal{I}}(d, C)) \leqslant n\} \\ (\geqslant n\ s.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \sharp(s^{\mathcal{I}}(d, C)) \geqslant n\} \end{aligned}$$

where $\sharp(M)$ denotes the cardinality of the set M and $s^{\mathcal{I}}(d, C)$ is defined as

$$\{d' \in \Delta^{\mathcal{I}} \mid (d, d') \in s^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}.$$

For $C, D \in \mathsf{con}(\mathcal{S})$, a *general concept inclusion* (GCI) is an expression $C \sqsubseteq D$. A finite set of GCIs is called a *TBox*. An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and a TBox $\mathcal{T}$ if it satisfies each GCI in $\mathcal{T}$.

An (ABox) *assertion* is an expression of the form $C(a)$, $r(a, b)$, $\neg r(a, b)$, or $a \neq b$, where $C \in \mathsf{con}(\mathcal{S})$ is a concept, $r \in \mathsf{rol}(\mathcal{S})$ is a role, and $a, b \in N_I$ are individual names. We call these assertions concept, role, negated role, and inequality assertions respectively. An *ABox* is a finite set of assertions. We use $\mathsf{Inds}(\mathcal{A})$ to

denote the set of individual names occurring in $\mathcal{A}$. An interpretation $\mathcal{I}$ *satisfies* an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $r(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, $\neg r(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$, and $a \not\approx b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* an ABox if it satisfies each assertion in $\mathcal{A}$, which we denote with $\mathcal{I} \models \mathcal{A}$.

Given a signature $\mathcal{S}$, a *knowledge base* $\mathcal{K}$ is a triple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ with $\mathcal{T}$ a TBox, $\mathcal{R}$ a role hierarchy, and $\mathcal{A}$ an ABox over $\mathcal{S}$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. We say that $\mathcal{I}$ *satisfies* $\mathcal{K}$ if $\mathcal{I}$ satisfies $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. In this case, we say that $\mathcal{I}$ is a *model* of $\mathcal{K}$ and write $\mathcal{I} \models \mathcal{K}$. We say that $\mathcal{K}$ *is consistent* if $\mathcal{K}$ has a model. △

The TBox and role hierarchy define a general schema and describe the concepts and their relations in the modelled domain. Sometimes this part is referred to as intensional knowledge. The ABox (partially) instantiates such a schema by adding assertions about individuals and this part is sometimes referred to as extensional knowledge.

A concept C is in *negation normal form* (NNF), if negation occurs only in front of atomic concepts. Any concept can be transformed in linear time into an equivalent one in NNF by pushing negation inwards, making use of de Morgan's laws and the duality between existential and universal restrictions, and between atmost and atleast number restrictions ($\leqslant n\ r.C$ and $\geqslant n\ r.C$ respectively) [68]. For a concept C, we use $\mathsf{nnf}(\neg C)$ to denote the NNF of $\neg C$.

The letter $\mathcal{S}$ in $\mathcal{SHOIQ}$ is an abbreviation for the DL $\mathcal{ALC}_{\mathcal{R}^+}$ [68], i.e., $\mathcal{ALC}$ extended with transitively closed roles. The letter $\mathcal{H}$ indicates the support of role inclusion axioms, $\mathcal{O}$ represents nominals, i.e., concepts such as $\{o\}$ for $o \in N_I$, $\mathcal{I}$ stands for inverse roles, and $\mathcal{N}$ ($\mathcal{Q}$) for (qualified) number restrictions of the form $\leqslant n\ s.\top$ and $\geqslant n\ s.\top$ ($\leqslant n\ s.C$ and $\geqslant n\ s.C$).

Please note that the term individuals is sometimes used ambiguously. It can refer to elements of the domain, i.e., for an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $d \in \Delta^{\mathcal{I}}$ is an individual in the domain of $\mathcal{I}$, but sometimes the term individual is also used in the context of ABoxes as an abbreviation for individual name or interpretation of an individual name. For example, when an ABox $\mathcal{A}$ contains the assertion $C(a)$, we might say that the individual a is an instance of the concept C instead of saying that the interpretation of the individual name a is in the extension of the concept C. It should, however, be clear from the context which of the two meanings is intended.

Some DL systems or algorithms make a *Unique Name Assumption* (UNA),

which means that different individual names are necessarily mapped to different elements of the domain. Without the UNA, different individual names can be mapped to the same element of the domain. In this case, it is usually possible to explicitly state that two individual names have to be mapped to different elements of the domain by allowing for inequality assertions in the ABox. It is easy to see that the UNA can then be simulated by adding inequality assertions for each pair of individual names.

2.2 Standard Reasoning Tasks

A DL system offers not only the possibility for storing knowledge, it also offers reasoning services for inferring implicit knowledge, determining concept satisfiability, i.e., if a concept can ever have instances, or for testing if a knowledge base is contradictory. Typically, the reasoners offer a query interface for these reasoning tasks and applications use this interface in order to support users in building knowledge bases that are consistent and for visualising the subsumption hierarchy to give users a quick overview of the modelled domain. A common inference task is, therefore, the computation of the subsumption hierarchy, i.e., a hierarchy based on the sub-class/super-class relationship between concepts in a knowledge base. Often these tests are performed w.r.t. a TBox or a knowledge base. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a knowledge base over a signature $\mathcal{S}$, C, D concepts over $\mathcal{S}$, and $a \in \mathsf{Inds}(\mathcal{A})$.

Knowledge Base Consistency Given $\mathcal{K}$ as input, a decision procedure for knowledge base consistency returns "$\mathcal{K}$ is consistent" if there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and it returns "$\mathcal{K}$ is inconsistent" otherwise.

Concept Satisfiability Given C and $\mathcal{K}$ as input, a decision procedure for concept satisfiability w.r.t. a knowledge base returns "C is satisfiable w.r.t. $\mathcal{K}$" if there is an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and an element $d \in \Delta^{\mathcal{I}}$ such that $\mathcal{I} \models \mathcal{K}$ and $d \in C^{\mathcal{I}}$ and it returns "C is unsatisfiable w.r.t. $\mathcal{K}$" otherwise.

Concept Subsumption Given C, D, and $\mathcal{K}$ as input, a decision procedure for concept subsumption w.r.t. a knowledge base, returns "D subsumes C w.r.t. $\mathcal{K}$" if there is no interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \mathcal{K}$ and there is an element

$d \in \Delta^{\mathcal{I}}, d \in C^{\mathcal{I}}$, and $d \in (\neg D)^{\mathcal{I}}$ and it returns "D does not subsume C w.r.t. $\mathcal{K}$" otherwise.

Instance Checking Given a, C, and $\mathcal{K}$, a decision procedure for instance checking, returns "a is an instance of C w.r.t. $\mathcal{K}$" if, for each interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \mathcal{K}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it returns "a is not an instance of C w.r.t. $\mathcal{K}$" otherwise.

Please note that all of the above reasoning tasks are formulated in terms of a decision problem, i.e., the answer could simply be reduced to yes/no or true/false. This has the advantage that we can study the behaviour of a given algorithm for the above problems in terms of complexity theory. A *computation problem*, such as instance retrieval, is usually reduced to its corresponding decision problem, e.g., in order to retrieve all instances of a concept C w.r.t. $\mathcal{K}$, we check, for each individual name a in $\mathcal{A}$, whether a is an instance of C and return the set of all those individual names for which this is the case. Such a procedure clearly terminates since an ABox contains finitely many individual names. We will later see that a similar relationship holds between the computation problem of query answering and its corresponding decision problem of query entailment.

Subsumption can be reduced to concept (un)satisfiability and vice versa. A concept D subsumes a concept C w.r.t. a knowledge base $\mathcal{K}$ iff the concept $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{K}$. A concept C is unsatisfiable w.r.t. $\mathcal{K}$ iff $\bot$ subsumes C w.r.t. $\mathcal{K}$. Concept satisfiability (and hence subsumption) can also be reduced to knowledge base consistency since C is satisfiable w.r.t. $\mathcal{K}$ iff $\mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{C(a_n)\})$ is consistent, where $a_n \in N_I$ is an individual name not occurring in $\mathcal{K}$. Instance checking can be reduced to knowledge base consistency checking as well since a is an instance of C iff $\mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{(\neg C)(a)\})$ is inconsistent. It is not hard to see that these reductions can be performed in linear time and, consequently, an algorithm that decides whether a knowledge base is consistent, can be used for all reasoning tasks described above.

In the presence of transitive roles and role hierarchies, a TBox $\mathcal{T}$ can be *internalised* using an "approximation" of a universal role u [6, 62, 68]. Testing the satisfiability of a concept w.r.t. a TBox and a role hierarchy can then be reduced to testing concept satisfiability w.r.t. the role hierarchy only. If the logic allows for nominals, an ABox can be expressed in terms of TBox axioms [115]. Together with the internalisation technique, we can thus reduce the task of

checking the consistency of a $\mathcal{SHOIQ}$ knowledge base to testing the satisfiability of a $\mathcal{SHOIQ}$-concept w.r.t. a role hierarchy.

Another inference service is called *entailment* or logical implication and allows to test if, for example, an ABox assertion or a TBox axiom is true in each model of a knowledge base. Conjunctive query entailment is a further generalisation since queries can contain variables and several concept and role expressions can be conjoined. Since this thesis is mainly concerned with conjunctive query entailment, we introduce this reasoning task in more detail in the following section.

Definition 2.4. Let $\mathcal{K}$ be a knowledge base and Γ a GCI or ABox assertion. We say that $\mathcal{K}$ *entails* Γ, written as $\mathcal{K} \models \Gamma$, if, for every interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \mathcal{K}$, it holds that

1. $a^{\mathcal{I}} \in C^{\mathcal{I}}$ if Γ is a concept assertion of the form $C(a)$,
2. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ if Γ is a role assertion of the form $r(a, b)$,
3. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$ if Γ is a negated role assertion of the form $\neg r(a, b)$,
4. $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if Γ is an inequality assertion of the form $a \neq b$, and
5. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ if Γ is a GCI of the form $C \sqsubseteq D$.

Otherwise $\mathcal{K}$ does not entail Γ and we write $\mathcal{K} \not\models \Gamma$. △

Please note that a knowledge base $\mathcal{K}$ is inconsistent iff $\mathcal{K} \models C(a)$ and $\mathcal{K} \models (\neg C)(a)$. For logics closed under negation, i.e., logics in which, for every concept C, the negation of C is expressible as well, checking concepts for satisfiability can be formulated in terms of entailment and vice versa. A concept C is satisfiable w.r.t. $\mathcal{K}$ iff $\mathcal{K} \not\models C \sqsubseteq \bot$. An axiom $C \sqsubseteq D$ is entailed by $\mathcal{K}$ iff $\mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{(C \sqcap \neg D)(a_n)\}$ is unsatisfiable for an individual name $a_n \in N_I$ not occurring in $\mathcal{K}$, and $\mathcal{K}$ entails a concept assertion $C(a)$ iff $\mathcal{K}' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\neg C(a)\})$ is unsatisfiable. The latter is known as the refutation theorem and builds the foundation of resolution and tableau based reasoning methods. Given a hypothesis such as $\mathcal{K} \models C(a)$, tableau methods try to prove that the hypothesis does not hold by establishing a counter-model, i.e., a model $\mathcal{I}$ of $\mathcal{K}$ in which a belongs to the extension of $\neg C$. Resolution based methods try to prove that the hypothesis holds by showing that the negated hypothesis leads to a contradiction. In case the procedures are complete, failure can be taken as proof of the opposite.

2.3 Conjunctive Queries

We now formally introduce the terms and notations for conjunctive queries used throughout this thesis. For query answering, the answer variables are often given in the head of the query. For example, in the introductory query from Section 1.2.2

$$(x, y, z) \leftarrow Father(x) \wedge Mother(y) \wedge hasChild(x, z) \wedge hasChild(y, z),$$

the query answers are those tuples (a_1, a_2, a_3) of individual names that, when substituted for x, y, and z respectively, result in a Boolean query that is entailed by the knowledge base. For simplicity, and since we mainly focus on query entailment, we do not use a query head even in the case of a non-Boolean query. Instead, we explicitly say which variables are distinguished/answer variables and which ones are non-distinguished/existentially quantified. Furthermore, we do not write a conjunctive query as a conjunction but as a set. For example, we write the Boolean version of the above query

$$() \leftarrow Father(x) \wedge Mother(y) \wedge hasChild(x, z) \wedge hasChild(y, z)$$

as

$$\{Father(x), Mother(y), hasChild(x, z), hasChild(y, z)\}.$$

The set notation allows for an easier definition of the query rewriting algorithm that we present in the following chapters. It further allows the use of standard set operations, e.g., we can define a sub-query simply as a subset of the original query.

We now give a more precise definition of Boolean conjunctive queries, followed by a definition of non-Boolean queries. We then show how non-Boolean queries can be reduced to Boolean queries. Finally, unions of conjunctive queries are introduced.

Definition 2.5. Let $\mathcal{L}$ be a Description Logic, $\mathcal{S} = (N_C, N_R, N_I)$ a signature, and N_V a countably infinite set of variables disjoint from N_C, N_R, and N_I. A *term* t is an element from $N_V \cup N_I$. Let C be an $\mathcal{L}$-concept, r an $\mathcal{L}$-role, and t, t' terms. A conjunct is an expression $C(t)$, $r(t, t')$, or $t \approx t'$ and we refer to these three different types of conjuncts as concept conjuncts, role conjuncts, and equality conjuncts respectively. A *conjunctive query* q is a non-empty set of conjuncts.

We use $\mathsf{Vars}(q)$ to denote the set of variables occurring in q, $\mathsf{Inds}(q)$ to denote the set of individual names occurring in q, and $\mathsf{Terms}(q)$ for the set of terms in q, where $\mathsf{Terms}(q) = \mathsf{Vars}(q) \cup \mathsf{Inds}(q)$. If all terms in q are individual names, we say that q is *ground*. A *sub-query* of q is simply a subset of q (including q itself).

Since equality is reflexive, symmetric and transitive, we define $\overset{*}{\approx}$ as the transitive, reflexive, and symmetric closure of $\approx$ over the terms in q. Hence, the relation $\overset{*}{\approx}$ is an *equivalence relation* over the terms in q, and we use $[t]_{\overset{*}{\approx}}$ for $t \in \mathsf{Terms}(q)$ to denote the $\overset{*}{\approx}$ *equivalence class of* t and, when $\overset{*}{\approx}$ is clear from the context, we omit the subscript and write just $[t]$.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. A total function $\pi\colon \mathsf{Terms}(q) \to \Delta^{\mathcal{I}}$ is an *evaluation* if (i) $\pi(a) = a^{\mathcal{I}}$ for each individual name $a \in \mathsf{Inds}(q)$ and (ii) $\pi(t) = \pi(t')$ for all $t \overset{*}{\approx} t'$. We write

- $\mathcal{I} \models^{\pi} C(t)$ if $\pi(t) \in C^{\mathcal{I}}$;
- $\mathcal{I} \models^{\pi} r(t, t')$ if $(\pi(t), \pi(t')) \in r^{\mathcal{I}}$;
- $\mathcal{I} \models^{\pi} t \approx t'$ if $\pi(t) = \pi(t')$.

If, for an evaluation π, $\mathcal{I} \models^{\pi} co$ for all conjuncts $co \in q$, we write $\mathcal{I} \models^{\pi} q$. We say that $\mathcal{I}$ *satisfies* q and write $\mathcal{I} \models q$ if there exists an evaluation π such that $\mathcal{I} \models^{\pi} q$. We call such a π a *match* for q in $\mathcal{I}$.

Let $\mathcal{K}$ be a knowledge base and q a conjunctive query. If, for every interpretation $\mathcal{I}$, $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models q$, we say that $\mathcal{K}$ *entails* q and write $\mathcal{K} \models q$.

For brevity and simplicity of notation, we define the relation $\bar{\in}$ over conjuncts in q as follows: $C(t) \;\bar{\in}\; q$ if there is a term $t' \in \mathsf{Terms}(q)$ such that $t \overset{*}{\approx} t'$ and $C(t') \in q$, and $r(t_1, t_2) \;\bar{\in}\; q$ if there are terms $t_1', t_2' \in \mathsf{Terms}(q)$ such that $t_1 \overset{*}{\approx} t_1'$, $t_2 \overset{*}{\approx} t_2'$, and $r(t_1', t_2') \in q$ or, in case the logic $\mathcal{L}$ supports inverses, $\mathsf{Inv}(r)(t_2', t_1') \in q$. △

Please note that the semantics we use here is the standard First-Order semantics that is also used for the standard reasoning tasks. The definition of the relation $\bar{\in}$ is clearly justified by definition of the semantics.

If we write that we replace $r(t, t') \;\bar{\in}\; q$ with $s(t_1, t_2), \ldots, s(t_{n-1}, t_n)$ for $t = t_1$ and $t' = t_n$, we mean that we first remove any occurrences of $r(\hat{t}, \hat{t}')$ and $\mathsf{Inv}(r)(\hat{t}', \hat{t})$ such that $\hat{t} \overset{*}{\approx} t$ and $\hat{t}' \overset{*}{\approx} t'$ from q, and then add the conjuncts $s(t_1, t_2), \ldots, s(t_{n-1}, t_n)$ to q.

Please note that the above definition of conjunctive queries allows for complex concepts that belong to the chosen Description Logic $\mathcal{L}$. For example, in a $\mathcal{SHOIQ}$ conjunctive query a concept conjuncts $C(t)$ is such that C is a $\mathcal{SHOIQ}$-concept. In case the logic $\mathcal{L}$ allows for inverses, inverse roles can also occur in the role conjuncts. Often conjunctive queries are defined such that only concept and role names can be used. Such a restriction is natural in the database setting and it is useful when studying the complexity of a query entailment algorithm. We comment on this in more detail in Section 2.4.

Without loss of generality, we assume that queries are connected. More precisely, let q be a conjunctive query. We say that q is *connected* if, for all $t \not\approx t' \in \mathsf{Terms}(q)$, there exists a sequence $t_1, \ldots, t_n$ such that $t_1 = t$, $t_n = t'$ and, for all $1 \leq i < n$, there exists a role r such that $r(t_i, t_{i+1}) \bar{\in} q$. Please note that we use the relation $\bar{\in}$ here, which implicitly uses the relation $\overset{*}{\approx}$ and abstracts from the directedness of role conjuncts. A collection $q_1, \ldots, q_k$ of queries is a *partitioning* of q if $q = q_1 \cup \ldots \cup q_k$, $\mathsf{Terms}(q_i) \cap \mathsf{Terms}(q_j) = \emptyset$ for $1 \leq i < j \leq k$, and each q_i is connected.

Lemma 2.6. *Let $\mathcal{K}$ be a knowledge base, q a conjunctive query, and $q_1, \ldots, q_k$ a partitioning of q. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_i$ for each i with $1 \leq i \leq k$.*

Informally, the above lemma is justified for the following reason: for a partitioning $q_1, \ldots, q_k$ of a query q, we can also write q in the following form using the standard notation of conjunction instead of our set notation:

$$\exists \vec{x_1} \mathsf{con}_1 \wedge \ldots \wedge \exists \vec{x_k} \mathsf{con}_k,$$

where, for each i with $1 \leq i \leq k$, con_i is the conjunction of all conjuncts in q_i and $\vec{x_i}$ contains all non-distinguished variables from q_i. The existential quantifiers can be written in front of each sub-formula con_i since the variable names are distinct in each sub-formula due to the definition of partitions. Thus, the sub-formulae do not interact with each other and such queries may be answered by considering one formula at a time.

A detailed proof is given in [129, 7.3.2] and, with this lemma, it is clear that the restriction to connected queries is indeed without loss of generality since entailment of q can be decided be checking entailment of each q_i in turn. In what follows, we therefore assume all queries to be connected.

We now define cyclic and acyclic queries by means of mappings to trees.

Definition 2.7. Let $\mathbb{N}^*$ be the set of all (finite) words over the alphabet $\mathbb{N}$. A *tree* T is a non-empty, prefix-closed subset of $\mathbb{N}^*$. The empty word ε is called the *root* of T. For $w, w' \in T$, we call w' a *successor* of w if $w' = w \cdot c$ for some $c \in \mathbb{N}$, where "$\cdot$" denotes concatenation. We call w' a *neighbour* of w if w' is a successor of w or vice versa.

For a mapping $f\colon A \to B$, we use $\mathsf{dom}(f)$ and $\mathsf{ran}(f)$ to denote f's *domain* A and *range* B, respectively. Given an equivalence relation $\stackrel{*}{\approx}$ on $\mathsf{dom}(f)$, we say that f is *injective modulo* $\stackrel{*}{\approx}$ if, for all $a, a' \in \mathsf{dom}(f)$, $f(a) = f(a')$ implies $a \stackrel{*}{\approx} a'$ and we say that f is *bijective modulo* $\stackrel{*}{\approx}$ if f is injective modulo $\stackrel{*}{\approx}$ and surjective.

A query q is *acyclic* if there exists a total function f from terms in q to a tree such that f is bijective modulo $\stackrel{*}{\approx}$ and $r(t, t') \bar{\in} q$ implies that $f(t)$ is a neighbour of $f(t')$; otherwise q is *cyclic*. △

Unions of conjunctive queries are a generalisation of conjunctive queries, which, intuitively, are simply a disjunction of conjunctive queries.

Definition 2.8. A *union of conjunctive queries* (UCQ) is a formula $q_1 \vee \ldots \vee q_\ell$, where each disjunct q_i is a conjunctive query.

A knowledge base $\mathcal{K}$ *entails* a union of conjunctive queries $q_1 \vee \ldots \vee q_\ell$, written as $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$, if, for each interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$, there is some i with $1 \leq i \leq \ell$ such that $\mathcal{I} \models q_i$. △

Without loss of generality (cf. [129, 7.3.3]) we assume that the variable names in each disjunct are different from the variable names in the other disjuncts. Since each disjunct is a conjunctive query that contains only non-distinguished (i.e., existentially quantified) variables this can always be achieved by naming variables apart [38]. We comment in the next section on the case when conjunctive queries also contain answer variables. We further assume that each disjunct is a connected conjunctive query. This is without loss of generality since a UCQ which contains unconnected disjuncts can always be transformed into conjunctive normal form; we can then decide entailment for each resulting conjunct separately and each conjunct is a union of connected conjunctive queries. Next, we describe this transformation in more detail and, for convenience, we write a conjunctive query $\{co_1, \ldots, co_k\}$ as $co_1 \wedge \ldots \wedge co_k$.

Lemma 2.9. *Let $\mathcal{K}$ be a knowledge base, $q = q_1 \vee \ldots \vee q_n$ a union of conjunctive queries such that, for each i with $1 \leq i \leq n$, $q_i^1, \ldots, q_i^{k_i}$ is a partitioning of the*

conjunctive query q_i. *Then* $\mathcal{K} \models q$ *iff*

$$\mathcal{K} \models \bigwedge_{(i_1,\ldots,i_n)\in\{1,\ldots,k_1\}\times\ldots\times\{1,\ldots,k_n\}} (q_1^{i_1} \vee \ldots \vee q_n^{i_n}).$$

Again, a detailed proof is given in [129, 7.3.3]. Please note that, due to the transformation into conjunctive normal form, the resulting number of unions of connected conjunctive queries for which we have to test entailment can be exponential in the size of the original query. When analysing the complexity of the decision procedures presented in Chapter 4 and Chapter 5, we show that the assumption that each CQ in a UCQ is connected does not increase the complexity.

2.3.1 Query Answering

We now make the connection between query entailment and query answering clearer. For query answering, let the variables of a conjunctive query be typed: each variable can either be non-distinguished (and thus implicitly existentially quantified) or distinguished. Let q be a conjunctive query in n variables (i.e., $\sharp(\mathsf{Vars}(q)) = n$), of which $v_1, \ldots, v_m$ ($m \leq n$) are distinguished variables (also called answer variables). We say that q has arity m. Let $\mathsf{Inds}_\mathcal{K}$ be the set of individual names from N_I that occur in $\mathcal{K}$ either in the form of an ABox assertion or as a nominal in case the considered logic allows for nominals. The *certain answers* of $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ to q are those m-tuples $(a_1, \ldots, a_m) \in \mathsf{Inds}_\mathcal{K}^m$ such that, for all models $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I} \models^\pi q$ for some π that satisfies $\pi(v_i) = {a_i}^\mathcal{I}$ for all i with $1 \leq i \leq m$. Hence, for the semantics of conjunctive queries in DL knowledge bases we adapt the well-known notion of certain answers in indefinite databases (see, e.g., [136]).

It is not hard to see that the answers of $\mathcal{K}$ to q can be computed by testing, for each $(a_1, \ldots, a_m) \in \mathsf{Inds}_\mathcal{K}^m$, whether the query q' obtained from q by replacing each occurrence of v_i with a_i for $1 \leq i \leq m$ is entailed by $\mathcal{K}$. Given that there are a finite number of individual names used in $\mathcal{K}$, there is a bound on the number of Boolean queries necessary to answer a non-Boolean query. This is clearly not very efficient, but optimisations can be used, e.g., to identify a (hopefully small) set of candidate tuples. Ideas on how to obtain suitable candidate sets are proposed, for example, by Sirin and Parsia [122] or Glimm [41].

Let $q = q_1 \vee \ldots \vee q_k$ be a union of conjunctive queries. For query answering over unions of conjunctive queries we assume that the answer variables occur in

each disjunct, i.e., if $v_1, \ldots, v_m$ are the answer variables, then we assume that $v_1, \ldots, v_m \in \mathsf{Vars}(q_i)$ for each i with $1 \leq i \leq k$.

2.3.2 Conjunctive Queries in Databases

Conjunctive queries are a standard means for querying databases. The Closed World Assumption , which is typically made in a database setting, means that we have complete knowledge of the domain and query entailment essentially corresponds to model checking. In our context, we have the Open World Assumption and, thus, incomplete information about the domain. There are, however, several approaches that have been developed for querying databases with incomplete information (see, for example, [56, 107, 137]).

A database schema consists of a set of constraints. We can see a schema also as a TBox $\mathcal{T}$ and, possibly, a role hierarchy $\mathcal{R}$ containing constraints that are written in a Description Logic. Given such a schema $\mathcal{S}$, an incomplete database $\mathcal{D}$ is a set of concept and role assertions over $\mathcal{S}$ and corresponds, essentially, to an ABox over $\mathcal{S}$. With incomplete information, the constraints in $\mathcal{S}$ have to be taken into account for query answering, whereas, if we assume complete information, the constraints do not play any role once we know that the data is consistent with the constraints. Intuitively, each interpretation $\mathcal{I}$ that is a model of both the schema $\mathcal{S}$ and the data $\mathcal{D}$ represents a complete database that is coherent with $\mathcal{D}$ and the DL constraints in $\mathcal{S}$.

With incomplete information, query answering w.r.t. a schema is the same as query answering in our setting. The only difference is that the considered schema language in databases is not as expressive as the logics we focus on here.

2.3.3 Why Conjunctive Queries

Conjunctive queries are very well known in the database community and familiar to many in the form of SQL queries. CQs are, therefore, clearly one of the candidates, when studying query languages for DL knowledge bases. We could, however, also allow different types of conjuncts, e.g., inequality conjuncts, or choose a more expressive query language such as regular path queries. Regular path queries are a further generalisation of unions of conjunctive queries and recent results show that for such queries the query entailment problem in $\mathcal{SHIQ}$ is decidable as well [30].

Since, for very expressive DLs, it was an open question whether CQs are decidable or not, the limitation to standard CQs as we define them above seems like a good starting point, and it was known that CQs and UCQs are decidable in $\mathcal{ALCNR}$ [86].

Recent results also show that several other extensions of CQs easily lead to undecidability. Rosati [109] shows that an extension of conjunctive queries with inequalities is undecidable even for the very basic DL $\mathcal{AL}$, although conjunctive queries (but not unions of CQs) are still decidable and computationally not harder than CQs without inequalities in the DL $\mathcal{EL}$ [22]. Rosati has also studied an extension of CQs and UCQs with safe negation, in which each conjunct of a CQ is either an atom or a negated atom. This allows, in particular, the use of negated role names in role conjuncts. The negation is used safely if, in each conjunct, each variable occurs in at least one positive atom. Deciding CQ entailment with safe negation is undecidable already for $\mathcal{AL}$. Furthermore, Rosati studies UCQs with universally quantified negation, where each CQ can contain conjuncts with negated atoms and in which the variables that only appear in negated atoms are universally quantified. Entailment for UCQs with universally quantification is, however, undecidable in every DL.

2.4 Combined and Data Complexity

An interesting question, when devising an algorithm, is its behaviour for different inputs, i.e., whether it always terminates and whether the time required to solve the problem is linear, polynomial, exponential, or even worse with respect to the size of the input. By analysing the computational complexity of an algorithm, one can, in particular, characterise the worst-case behaviour of an algorithm. We give just a brief introduction into the areas of complexity theory that are of relevance for this thesis and refer interested readers to Papadimitriou's textbook [101].

For the complexity analysis, we distinguish between combined and data complexity [138]. For the standard reasoning tasks, e.g., knowledge base consistency, the combined complexity is measured in the size of the input knowledge base. For query entailment, the size of the query is additionally taken into account. The *size* of a knowledge base $\mathcal{K}$ or a query q is simply the number of symbols needed to write it over the alphabet of constructors, concept, role, individual, and variable names that occur in $\mathcal{K}$ or q, where numbers are encoded in binary

(if not mentioned otherwise). When analysing the data complexity of a problem, we consider the ABox as the only input for the algorithm, i.e., the size of the TBox, the role hierarchy, and the query is fixed. When the size of the TBox, role hierarchy, and query is small compared to the size of the ABox, the data complexity of a reasoning problem is a more useful performance estimate since it tells us how the algorithm behaves when the number of assertions in the ABox increases.

For the complexity analysis, we assume that all concepts in concept conjuncts or ABox assertions are literals, where a literal is either a concept name or a negated concept name. If the input query or ABox contains concept conjuncts or assertions in which the concept is not a literal, we can easily transform these into literal ones in an entailment preserving way: for each concept conjunct $C(t)$ in the query with C a non-literal concept, we introduce a new atomic concept $A_C \in N_C$, add the axiom $C \sqsubseteq A_C$ to the TBox, and replace $C(t)$ with $A_C(t)$; for each non-literal concept assertion $C(a)$ in the ABox, we introduce a new atomic concept $A_C \in N_C$, add an axiom $A_C \sqsubseteq C$ to the TBox, and replace $C(a)$ with $A_C(a)$. This transformation is obviously polynomial, so without loss of generality, we can assume that the ABox and query contain only literal concepts. This has the advantage that the size of each conjunct and ABox assertion is constant. In our definition of conjunctive queries, we allow for complex concepts since the query rewriting algorithms that we introduce in Chapter 4 and Chapter 5 might produce such queries even if the input query uses only atomic concepts and roles.

The computational complexity of the standard reasoning problems such knowledge base consistency checking are well-understood, and it is known that the combined complexity of these reasoning problems is ExpTime-complete for $\mathcal{SHIQ}$ [132] and NExpTime-complete for $\mathcal{SHOIQ}$ [132]. For $\mathcal{SHOQ}$ an ExpTime upper bound was always conjectured, but a proof has, to the best of our knowledge, never been published. As part of our conjunctive query entailment decision procedure for $\mathcal{SHOQ}$ presented in Chapter 5, we devise an ExpTime decision procedure for $\mathcal{SHOQ}$ knowledge base consistency checking, thereby showing that this problem is indeed ExpTime-complete for $\mathcal{SHOQ}$. For $\mathcal{SHIQ}$, instance retrieval is known to be data complete for co-NP [75]. Please note that, as usual, when saying that a reasoning problem, e.g., instance retrieval, is in a complexity class $\mathcal{C}$, we mean that the decision problem corresponding to the stated task (e.g., instance checking in the case of instance retrieval) is in $\mathcal{C}$.

Chapter 3

Related Work and Alternative Approaches

In this chapter, we introduce related work, existing approaches, and areas that have a close connection with query entailment. We also point out why some of the existing approaches are not decision procedures, in contrary to what is claimed, or cannot easily be applied to expressive logics. We start with the introduction of recently developed related approaches.

3.1 Conjunctive Queries for Expressive Description Logics

Very recently, an automata-based decision procedure for positive existential path queries over $\mathcal{ALCQI}b_{reg}$ knowledge bases has been presented [30]. Positive existential path queries generalise unions of conjunctive queries and since a $\mathcal{SHIQ}$ knowledge base can be polynomially reduced to an $\mathcal{ALCQI}b_{reg}$ knowledge base, the presented algorithm is a decision procedure for (union of) conjunctive query entailment in $\mathcal{SHIQ}$ as well. The automata-based technique can be considered more elegant than the rewriting algorithm that we present in Chapter 4, but it does not give an NP upper bound for the data complexity as our technique.

Most existing algorithms for conjunctive query answering in expressive DLs assume, however, that role atoms in conjunctive queries use only roles that are not transitive. Under this restriction, decision procedures for various DLs around $\mathcal{SHIQ}$ are known [64, 97, 129], and it is known that answering conjunctive queries

in this setting is data complete for co-NP [97]. Another common restriction is that only individuals named in the ABox are considered for the assignments of variables. In this setting, the semantics of queries is no longer the standard First-Order one. It is not hard to see that conjunctive query answering with this restriction can be reduced to standard instance retrieval by replacing the variables with individual names from the ABox and then testing the entailment of each conjunct separately. Most of the implemented DL reasoners, e.g., KAON2, Pellet, and RacerPro, provide an interface for conjunctive query answering in this setting and employ several optimisations to improve the performance [93, 122, 143]. Pellet appears to be the only reasoner that also supports the standard First-Order semantics for $\mathcal{SHIQ}$ conjunctive queries under the restriction that the queries are acyclic.

To the best of our knowledge, it is still an open problem whether conjunctive query entailment is decidable in $\mathcal{SHOIQ}$. Regarding undecidability results, it is known that conjunctive query entailment in the two variable fragment of First-Order Logic $\mathcal{L}_2$ is undecidable [109] and Rosati identifies a relatively small set of constructors that causes the undecidability.

3.2 Conjunctive Queries for Tractable Description Logics

Recently, query entailment and answering have been extensively studied for logics that are tractable, i.e., logics that have reasoning problems of at most polynomial complexity. For example, the constructors provided by logics of the DL-Lite family [5, 27, 28] are chosen such that the standard reasoning tasks are PTime-complete and query entailment is in LogSpace with respect to data complexity, whereas we consider logics in which these problems are ExpTime-complete and NP-complete respectively. Furthermore, for logics such as the ones in the DL-Lite family, TBox reasoning can usually be done independently of the ABox, which means that the ABox can be stored and accessed using a standard database SQL engine. An efficient implementation for conjunctive query answering over DL-Lite knowledge bases is, for example, available with the QuOnto system. QuOnto uses a database to store the ABox and reduces CQs to standard SQL queries.

Conjunctive queries have also been studied for the DL $\mathcal{EL}$, which is a logic that has PTime-complete standard reasoning problems. It was shown that UCQ

entailment in $\mathcal{EL}$ and in its extensions with role hierarchies is NP-complete regarding the combined complexity [108]. The data complexity of UCQ entailment in $\mathcal{EL}$ is PTIME-complete [26, 109]. Allowing, additionally, role composition in the logic as in $\mathcal{EL}^+$, leads to undecidability [82, 83, 108].

Several results have also been obtained in the context of databases with incomplete information [56, 107, 137]. In this setting, DLs can be used as schema languages, but the expressivity of the considered DLs is usually much lower than the expressivity of $\mathcal{SHIQ}$ or $\mathcal{SHOQ}$. Due to the lower expressivity of the logics, the used techniques cannot directly be applied in our setting.

3.3 Modal Correspondence Theory

Some interesting work regarding query answering and query entailment has also been done using modal correspondence theory [14, 81, 111]. Correspondence theory is devoted to the question whether a modally definable class of frames (e.g., reflexive frames) can also be defined by a First-Order Logic formula and, if so, whether the corresponding FOL formula can be found efficiently (and similarly in the other direction).

It is well-known that many Description Logics are notational variants of Modal Logics. Hence, many results obtained for Modal Logics can directly be applied to Description Logics. Zolin [149, 150] shows how certain classes of Boolean and unary conjunctive queries (which are FOL formulae) can be answered by computing, in time linear in the size of the query, a modal logic formula (and, hence, a DL concept) that corresponds to the query. In case of unary queries, i.e., queries with one answer variable, the query answers are exactly those individual names that are instances of the computed concept. In case of Boolean queries, the query is entailed by the knowledge base iff the concept is satisfiable. Since the concepts can be computed in linear time, the complexity of answering CQs of the supported classes is computationally not harder than the standard reasoning tasks.

The classes of conjunctive queries that can be answered with this technique contain Boolean and unary conjunctive queries that are either tree-shaped (in which case the standard rolling-up technique is used) or in which the queries contain cycles of particular shapes. We illustrate this technique by means of an example and refer interested readers for more examples and for a detailed

description of the supported classes to Zolin [149, 150]. An interesting property of the technique is that it is independent of the considered logic or knowledge base. Queries that fall into the supported classes can, therefore, be answered w.r.t. any knowledge base in any DL from $\mathcal{ALC}$ upwards.

For an example, let $\mathcal{K}$ be a knowledge base in any DL from $\mathcal{ALC}$ upwards, $X \in N_C$ an atomic concept that does *not* occur in $\mathcal{K}$, r a role from $\mathcal{K}$, and $a \in N_I$ an individual name/nominal occurring in $\mathcal{K}$. Zolin shows that the task of deciding whether $\mathcal{K} \models \{(\neg X \sqcup \exists r.X)(a)\}$ is equivalent to checking whether a is an r-successor of itself, i.e., whether $(a, a) \in r^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. Please note that it is essential that X is a fresh concept.

In order to find *all* individual names $a_i \in \mathsf{Inds}(\mathcal{A})$ such that $(a_i, a_i) \in r^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{K}$, we can, for each individual $a_i \in \mathsf{Inds}(\mathcal{A})$, check whether $\mathcal{K} \models \{(\neg X \sqcup \exists r.X)(a_i)\}$. It is not hard to see that we can equivalently use the conjunctive query $\{r(x, x)\}$, where x is an answer variable.

In (multi-)Modal Logic syntax, we can write the DL concept $\neg X \sqcup \exists r.X$ as $X \rightarrow \Diamond_r X$, for a propositional variable X. Assuming only a single relation r, we can simplify this to $X \rightarrow \Diamond X$. This formula is well known in Modal Logics for expressing reflexivity (mostly written as $p \rightarrow \Diamond p$).

In general, the proposed algorithm takes a query q of the supported classes as input and invokes an algorithm from Correspondence Theory to build a modal formula ϕ that (locally) corresponds to the query. It then translates ϕ into a DL concept C_q (by using fresh concept names). The concept C_q is then used to answer the query. Since CQ entailment in $\mathcal{SHIQ}$ is 2ExpTime-hard [88], whereas instance checking is in ExpTime [132], it is clear that such a reduction cannot work in the general case. The technique could, however, be valuable as an optimisation for the supported query classes.

3.4 Query Containment

A related reasoning problem to query entailment is *query containment*. Query containment is the problem of checking whether, for every ABox, the result of one query is a subset of the result of another query. A *schema* is a set of constraints and in our setting a schema can be seen as a TBox plus a role hierarchy. Query containment w.r.t. a schema is the problem of checking whether, for every ABox and every model of the schema and the ABox, the result of one query is a subset

of the result of another query. More precisely, a query q is contained in a query q' w.r.t. a schema $\mathbf{S}$, denoted as $\mathbf{S} \models q \sqsubseteq q'$, if, for every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathbf{S}$ and $\mathcal{I} \models q$, it holds that $\mathcal{I} \models q'$.

Query containment is important in many areas, including information integration [23, 24, 31] and query optimisation [2, 3]. It is well known in the database community that there is a tight connection between the problems of conjunctive query containment and conjunctive query answering [32] and this is also the case for query containment w.r.t. constraints specified in a schema. Due to this close relationship, query containment w.r.t. a TBox and RBox can be reduced to deciding query entailment for (unions of) conjunctive queries w.r.t. a knowledge base and vice versa [22, 29].

In order to reduce query answering to query containment, we can proceed as follows: let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a knowledge base and q a union of conjunctive queries with answer variables $\vec{v} = v_1, \ldots, v_m$. In order to decide whether a tuple $\vec{a} = a_1, \ldots, a_m$ is an answer for q, we define the following two Boolean queries $q_{\mathcal{A}}$ and $q_{\vec{a}}$:

$$q_{\mathcal{A}} = \{C(a) \mid C(a) \in \mathcal{A}\} \cup \{r(a, b) \mid r(a, b) \in \mathcal{A}\}$$

and $q_{\vec{a}}$ is the query obtained from q by replacing, for each i with $1 \leq i \leq m$, each occurrence of v_i in q with a_i. The first query encodes, intuitively, the facts given in the ABox. Then $\vec{a}$ is an answer for q w.r.t. $\mathcal{K}$ iff $(\mathcal{T}, \mathcal{R}) \models q_{\mathcal{A}} \sqsubseteq q_{\vec{a}}$. The correctness of this reduction is quite immediate from the given semantics and a detailed proof is given by Abiteboul and Duschka [1].

We now give an example of how query containment can be reduced to query entailment. For simplicity of the example, we assume that both queries are conjunctive queries and not unions of conjunctive queries. For more details and proofs of the general case (where we have to handle disjunctive ABoxes), we refer interested readers to Calvanese et al. [22, 29] and Horrocks et al. [65, 67]. Let $\mathcal{T}$ be a TBox, $\mathcal{R}$ a role hierarchy, and q and q' two conjunctive queries of the same arity. We implicitly assume that the names of the non-distinguished variables in q and q' are different, which is without loss of generality because arbitrary names can be chosen without affecting the results of the query. We further assume that the answer variables and individual names/constants occur in both queries. If a name a is missing in one of the queries, we can always add the conjunct $(\top)(a)$. Let $x_1, \ldots, x_m$ be the answer variables. In order to reduce to problem of deciding whether $(\mathcal{T}, \mathcal{R}) \models q \sqsubseteq q'$ to query entailment, we define the ABox $\mathcal{A}_q$ as the

ABox obtained from q by treating each variable name as an individual name and by treating $x_1, \ldots, x_m$ as individual names instead of as answer variables in q', which makes q' a Boolean conjunctive query. We then have that $(\mathcal{T}, \mathcal{R}) \models q \sqsubseteq q'$ iff $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}_q) \models q'$. The correctness of this claim follows quite immediately from the definition of the semantics and the definition of $\mathcal{A}_q$ from q.

3.4.1 The Difficulty of Regular Expressions

The first conjunctive query algorithm [22] was actually specified for the purpose of deciding conjunctive query containment for $\mathcal{DLR}_{reg}$ [35]. $\mathcal{DLR}_{reg}$ is a very expressive Description Logic that allows for n-ary relations (i.e., a generalisation of roles), regular expressions over roles, and role hierarchies. Compared to $\mathcal{SHIQ}$, $\mathcal{DLR}_{reg}$ does not allow the specification of roles as transitive, but regular expressions in concepts can be used to express the transitive closure of the role. Inverse roles can also only be expressed in concepts via regular expressions and they cannot be used in the role hierarchy, e.g., to state that a role is symmetric. On the other hand, the regular expressions over roles such as role composition, disjunction, or transitive closure, make $\mathcal{DLR}_{reg}$ quite powerful.

The presented algorithm contains one of the fundamental ideas for query entailment that is also used in this thesis, namely, the reduction of tree-shaped queries to concepts. The equivalence of such queries with concepts was already known from First-Order Logic [15], but Calvanese et al. were the first to use this equivalence for the purpose of query answering.

Contrary to what is claimed, the presented algorithm is not a decision procedure since it is incomplete for cyclic queries. This has already been pointed out by Horrocks et al. [65] since, in some cases, a cyclic query can be transformed into a tree-shaped query by identifying variables (cf. Example 1.2 and Figure 1.3 in the introduction). We show here that this is not the only source of incompleteness and that just identifying variables in all possible ways is not enough to obtain a decision procedure. As a consequence of this observation, the journal version of the above mentioned $\mathcal{DLR}_{reg}$ algorithm [29] allows for a less expressive query language compared to the original paper.

The main problem is that the algorithm depends on a strict tree model property, which can be disturbed by transitive or inverse roles. We give an example in standard Description Logic notation, which is more intuitive and easily possible since we need only binary relations and regular expressions that correspond to

inverse roles. The $\mathcal{DLR}_{reg}$ notation is slightly different and the algorithm involves a translation into graded converse Propositional Dynamic Logic (CPDL_g) that results in quite lengthy formulae.

As an example for the incompleteness in case of cyclic queries, consider the Boolean conjunctive query $q = \{(\exists r.\top)(a)\}$ with $a \in \mathsf{Inds}(q)$, the cyclic Boolean conjunctive query $q' = \{r(z_1, z_2), s^-(z_2, z_1)\}$ with $z_1, z_2 \in \mathsf{Vars}(q')$, and the schema $(\mathcal{T}, \mathcal{R})$ with $\mathcal{T} = \emptyset$ and $\mathcal{R} = \{r \sqsubseteq s\}$. Clearly, every model for $(\mathcal{T}, \mathcal{R})$ and q is also a model of q' and, hence, $(\mathcal{T}, \mathcal{R}) \models q \sqsubseteq q'$. Calvanese et al. argue, however, that cycles cannot be expressed in the schema itself and, therefore, the variables in q' have to be replaced with individual names or variable names from q. In our example, this is only a, resulting in $q' = \{r(a, a), s^-(a, a)\}$. After the replacement it is, however, no longer the case that every model of q is also a model of q' and, as a result, the algorithm incorrectly answers $(\mathcal{T}, \mathcal{R}) \not\models q \sqsubseteq q'$.

It should be noted that due to the role axiom $r \sqsubseteq s$ and due to the semantics of inverse roles the simpler query $q'' = \{r(z_1, z_2)\}$ is entailed by the schema if and only if q' is entailed by the schema. Hence, both queries are equivalent. Clearly, the query q'' is acyclic, which illustrates again why replacing the variables with constants is incorrect.

In order to devise a decision procedure for query containment in $\mathcal{DLR}_{reg}$, one would need to analyse cyclic queries more carefully. It is obvious that, as in the given example, not every cycle constitutes a "real" cycle that can only be present in each model of the schema due to ABox assertions. Please note also that such situations arise only in the presence of regular expressions in the query q' and that the algorithm is a decision procedure for conjunctive query entailment in $\mathcal{DLR}$ (i.e., $\mathcal{DLR}_{reg}$ without regular expressions).

3.5 Rule Formalisms

Function-free Horn rules [60, 87, 93, 105] are, like Description Logics, a decidable yet very expressive subset of First-Order Logic. Roughly speaking, a rule is a statement of the following form: if the precondition P holds, then the conclusion C holds, where P and C are logical formulae. As an example, consider the following rule which, intuitively, expresses the requirements for concluding that

y is the aunt of x:

$$hasAunt(x, y) \leftarrow hasParent(x, z) \wedge hasSibling(z, y) \wedge Female(y).$$

In general, the expression on the left hand side of the arrow is called the *head* of the rule and the right hand side is called the rule *body*. A finite set of rules is called a *program*. A set of rules is called *recursive* if the body of one rule directly or indirectly depends on the head of another rule and it is called *non-recursive* otherwise. Unlike typical Description Logics, rules also allow for expressing non-tree-like structures (as in the above example) by allowing for an arbitrary interaction between variables in a rule. Predicates are also not limited to unary and binary ones, but can be of any arity. Many rule formalisms also allow for using negation-as-failure, which introduces a form of non-monotonic behaviour. A combination of DL knowledge bases with rules is, therefore, desirable for many applications, but an unrestricted combination of both formalisms easily leads to the undecidability of the standard reasoning tasks [59, 87]. In order to provide a decision procedure for rules, it is, therefore, necessary to impose suitable restrictions on the set of allowed rules.

When combining a DL knowledge base with rules, the predicates of a rule, e.g., $Female$ or $hasAunt$, may not necessarily appear in the DL part of the knowledge base. If the predicate is a concept or role name from the signature of the knowledge base, the atom is called a *DL-atom*, otherwise it is a *non-DL-atom*. Conjunctive queries can be seen as a special case of such rules: the rule body corresponds to a conjunctive query, while the head corresponds to the tuple of answer variables. A decision procedure for consistency of DL knowledge bases extended with non-recursive rules is, therefore, also a decision procedure for conjunctive query entailment. For deciding query entailment, the rules can be restricted even further to rules where the head is a non-DL-atom and all atoms in the body are DL-atoms. Going back to the previous example, we can query for all pairs of individuals (a, b) for which b is the aunt of a, by extending the knowledge base again with the rule

$$hasAunt(x, y) \leftarrow hasParent(x, z) \wedge hasSibling(z, y) \wedge Female(y);$$

this time assuming that $hasParent$, $hasSibling$, and $Female$ are DL-atoms. The query answers are then all those pairs (a, b) for which adding $\neg hasAunt(a, b)$ leads

to an inconsistency.

There are several different decidable rule formalisms known, and decidability is commonly achieved by imposing some form of safety condition on the rules [87, 93, 106]. One possible safety condition [93] requires that each variable occurs in at least one non-DL-atom in the rule body. Intuitively, forcing variables to occur in non-DL-atoms assures that all variables are bound to individual names occurring in the ABox of the knowledge base. Without such safety conditions, one has to significantly reduce the expressivity of the Description Logic and Levy and Rousset [87] show that allowing either universal quantification or (even unqualified) number restrictions leads to undecidability.

Relaxed safety conditions also allow for existentially quantified variables, i.e., variables that do not necessarily correspond to individuals named in the ABox. A recent example is Rosati's $\mathcal{DL}$+*log* framework [106]. In this framework, a knowledge base in the Description Logic $\mathcal{DL}$ can be extended with so called weakly-safe Datalog rules. The weakened safety condition ensures, intuitively, that at least the variables in the head of a rule correspond to named individuals from the ABox, which results in a close similarity between a rule and a conjunctive query. Indeed, Rosati shows that the consistency of such extended knowledge bases is decidable iff the conjunctive query containment problem for $\mathcal{DL}$ is decidable. Since we present decision procedures for conjunctive queries in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ in the following two chapters, we close the open problem of whether the consistency of $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ knowledge bases extended with weakly-safe rules is decidable.

3.5.1 The Carin System

The first framework for combining a Description Logic knowledge base with rules was the CARIN system [86]. CARIN provides a so-called *existential entailment* algorithm that is a decision procedure for non-recursive rules and conjunctive queries over $\mathcal{ALCNR}$ knowledge bases (where $\mathcal{R}$ stands for role conjunction). The algorithm is very similar to the known tableau algorithms that are used in many modern reasoning systems, although the original algorithm is specified in terms of a constraint system. We introduce the basic ideas of the algorithm here, but we use the nowadays more standard terminology of tableau algorithms. We also briefly illustrate how this algorithm can be extended to $\mathcal{SHIQ}$ if the query contains only simple roles, and the problem that arises when we, additionally, allow for either nominals or transitive roles in the query.

In order to decide conjunctive query entailment, the algorithm constructs completion trees/graphs for the knowledge base. For simpler Description Logics such as $\mathcal{ALCNR}$, there is a direct correspondence between the completion graphs and models of the knowledge base. The model that we can build from a complete and clash-free completion graph is called a *canonical model*. We give a more precise definition for canonical models of $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ knowledge bases in Chapter 4 and Chapter 5 respectively. Initially, a completion graph contains a node for each individual in the ABox and the nodes are labelled with the concepts from the corresponding concept assertions in the ABox. Each role assertion is represented by an edge between two nodes in the initial completion graph. An initial completion graph is expanded according to a set of *expansion rules* that reflect the constructors allowed in the language. The expansion stops when an obvious contradiction, called a *clash*, occurs, or when no more rules are applicable. In the latter case, the completion graph is called *complete*. Termination is guaranteed by a cycle-checking technique called *blocking*. The expansion and blocking rules are such that we can build a model for the knowledge base from each complete and clash-free completion graph.

It should be noted that $\mathcal{ALCNR}$ has the finite model property and, in particular, it is possible to construct a finite canonical model from each complete and clash-free completion graph. The algorithm uses the complete and clash-free completion graphs to syntactically check whether a match for the given query exists. For a match, each term that occurs in a concept conjunct $C(t)$ must be mapped to a node in the completion graph that has C in its label. Each pair of terms (t, t') that occurs in a role conjunct $r(t, t')$ must be mapped to a pair of nodes (n, n') such that n' is an r-successor of n. Each pair of terms (t, t') for which the query contains an equality conjunct $t \approx t'$ must be mapped to the same node in the completion graph and each constant a that occurs in the query must be mapped to the node in the completion graph that represents a. Given an $\mathcal{ALCNR}$ knowledge base $\mathcal{K}$ and a Boolean conjunctive query q, the algorithm answers "$\mathcal{K}$ entails q" if a match for q exists in each complete and clash-free completion graph and it answers "$\mathcal{K}$ does not entail q" otherwise.

There are two main problems that need to be overcome in order to use this approach. In the first place, we need to (syntactically) check in the completion graph whether the model represented by the completion graph satisfies the query. This might be difficult if a non-atomic concept C is used in one of the concept

conjuncts of the query because an element of the model might be in the extension of C without this being explicitly stated in the completion graph. This problem is overcome by augmenting the knowledge base with axioms of the form $\top \sqsubseteq C \sqcup \neg C$ for each concept C that occurs in a concept conjunct in q. This clearly has no logical impact on the knowledge base because $C \sqcup \neg C$ is equivalent to $\top$, but it ensures that, for each node in the completion graph, a decision is made as to whether the corresponding domain element belongs to the extension of C or to the extension of $\neg C$.

The second problem is that, in general, a knowledge base can have infinitely many possibly infinite models, whereas the algorithm constructs only a subset of the finite models of the knowledge base. Since the query entailment holds only if the query is true in all models of the knowledge base, we have to show that inspecting only a subset of the models, namely the canonical ones, suffices to decide query entailment. This problem is overcome by showing that, if there is a model in which q is false, then there is also a canonical model in which q is false. A major modification of the standard blocking technique is needed in order to ensure that this is the case. For $\mathcal{ALCNR}$ knowledge bases, the standard blocking condition is to stop the expansion of a branch in the constraint system if we find a pair of nodes x and y such that x is an ancestor of y and the label of y is a subset of the label of x. In this case, we say that x blocks y and no further expansion rules are applied to y. In the model for the completion graph, a block corresponds to a cyclic path that links the predecessor of the blocked node y to x instead of to y. For query entailment, the blocking condition additionally has to take into account the length of the longest path in the query. We illustrate this by means of an example.

Example 3.1. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ALCNR}$ knowledge base with $\mathcal{T} = \{C \sqsubseteq \exists r.C\}$ and $\mathcal{A} = \{C(a)\}$ and let $q = \{r(x, x)\}$ be a Boolean conjunctive query. It is not hard to check that $\mathcal{K} \not\models q$. Figure 3.1 illustrates the only (apart from differences in naming) complete and clash-free completion graph for $\mathcal{K}$ that the algorithm would construct. Figure 3.2 shows a representation of a corresponding canonical model $\mathcal{I}$. The loop in the model $\mathcal{I}$ occurs since the node n_1 blocks the node n_2 in the completion graph. It is not hard to check that $\mathcal{I} \models \mathcal{K}$ and that the mapping $\pi\colon x \mapsto n_1$ is such that $\mathcal{I} \models^{\pi} q$. Since the completion graph shown in Figure 3.1 is the only completion graph that the algorithm would construct, we would wrongly conclude that $\mathcal{K}$ entails q.

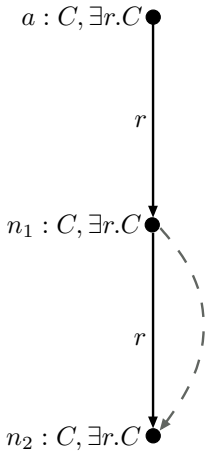

Figure 3.1: A complete and clash-free completion graph for $\mathcal{K}$. The node n_2 is blocked by the node n_1, which is indicated by the dashed line.

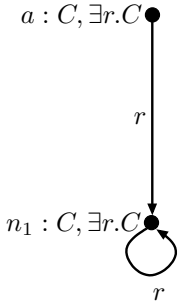

Figure 3.2: A graphical representation of a canonical model $\mathcal{I}$ for $\mathcal{K}$.

There are, however, other models of the knowledge base that are counter-models for the query entailment, i.e., we can find a model $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$, but $\mathcal{I} \not\models q$. One such example is shown in Figure 3.3. Another example consists of a single, infinite r-chain rooted in a. Intuitively, the model depicted in Figure 3.3 shows that we can still use finite models, but we have to make sure that the cycles that occur in the model due to blocking are "big enough", where "big enough" depends, intuitively, on the length of the longest path in the query.

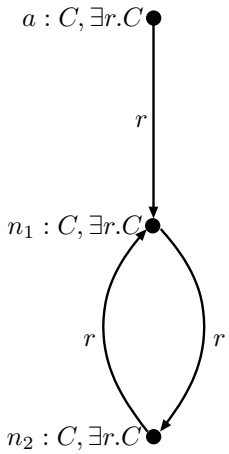

Figure 3.3: A graphical representation of a model $\mathcal{I}'$ for $\mathcal{K}$ that does not satisfy q.

Instead of using a pair of nodes x and y in the blocking definition such that x

is an ancestor of y and the label of y is a subset of the label of x, we want to make sure that the path from x to y is longer than any path in the query. Since the number of conjuncts in the query can clearly can be used as an upper approximation for the longest path in the query, the blocking condition in CARIN requires two isomorphic trees (instead of just two nodes) such that the depth of the trees is equal to the number of conjuncts in the query. The leaves of the descendant tree are then considered as blocked. In Example 3.1, the query contains only one conjunct. Hence, the revised blocking condition requires just two trees of depth one.

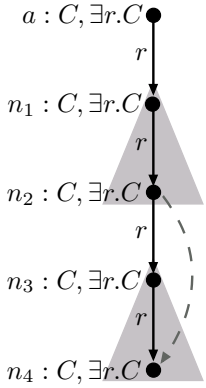

Figure 3.4: An abstraction of a complete and clash-free completion graph for $\mathcal{K}$ requiring two isomorphic trees of depth one in the blocking condition.

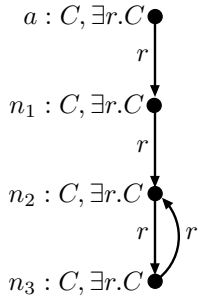

Figure 3.5: An abstraction of the model for the completion graph in Figure 3.4.

Figure 3.4 shows an abstraction of a complete and clash-free completion graph using the modified blocking condition. Please note that, due to the simplicity of the example, the trees consist of simple paths. In the canonical model that corresponds to the completion graph (see Figure 3.5), we have a cycle from the element that corresponds to the predecessor of the blocked node to the element that corresponds to the blocking node.

When building a model from a completion graph, one can, instead of building a cycle, also append infinitely many copies of the blocking tree and the path between the blocking and the blocked tree. In our example, this would result in a model that consists of an infinite r-chain. This technique is usually used for more expressive logics and also works for $\mathcal{ALCNR}$. If the logic does not allow for transitive roles, or the query contains only simple roles, the CARIN technique works for logics as expressive as $\mathcal{SHIQ}$ [97]. Intuitively, this can be proven in

two steps:

1. In the first step, we show that, for query entailment, we can restrict our attention to the canonical models of the knowledge base.

2. In the second step, we show that the query is satisfied in each complete and clash-free completion graph iff it is satisfied in each canonical model of the knowledge base.

Showing that the first claim holds is very useful for showing the correctness of any query entailment procedure and we give definitions of canonical models and detailed proofs for this claim in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ in the following two chapters (Lemma 4.2 and Lemma 5.3). The main idea of the proof is that we can take an arbitrary counter-model for the query, i.e., a model of the knowledge base that does not satisfy the query and "unravel" it into a canonical model such that the non-entailment of the query is preserved. This is possible since, during the unravelling process, cycles are replaced with infinite paths, while the interpretation of concepts is preserved. Intuitively, this "breaking" of cycles does not introduce any edges that could make the query true in the unravelled model. Once this claim is proved, we can restrict our attention to the canonical models of the knowledge base only.

For the second claim, we then have to show that the query is satisfied in each complete and clash-free completion graph iff it is satisfied in each canonical model of the knowledge base. The only-if direction is straightforward because a mapping for the query into the completion graph can directly be used in the corresponding unravelled canonical model. The if direction is not too hard to show given the assumption that the logic does not allow for transitive roles or that the query contains only simple roles. Under this restriction, we can take any canonical model of the knowledge base together with a match π for the query and construct a modified mapping π' that uses only those parts of the canonical model that correspond to the completion graph by "shifting" the mapping upwards to an isomorphic part that is closer to the roots of the canonical model. The roots, in this case, are those elements that correspond to named individuals.

3.5.2 Extensions of the Carin System

It should be noted that the technique of "shifting" the mapping up as suggested above does not work if the query contains non-simple roles [44]. Hence, the

proof does not easily extend to $\mathcal{SHIQ}$ with unrestricted conjunctive queries as claimed by Ortiz de la Fuente et al. [98] and later works [97] are indeed restricted to conjunctive queries that contain only simple roles. We illustrate the problem of non-simple roles here by means of an example. We try to show that the query is true in each canonical model of the knowledge base iff the query has a mapping into each complete and clash-free completion graph. The problematic direction is the only if direction, which, in its contrapositive form, claims the following: if there is a complete and clash-free completion graph such that the query has no mapping into the completion graph, then there is a canonical model for the knowledge base that does not satisfy the query. The proof technique that is used, is based on the idea that if the query has no mapping into a complete and clash-free completion graph, then the query is also false in the canonical model for that completion graph. In the following example, we show that this is not the case if the query contains transitive or non-simple roles. Other proof techniques might work, but have, to the best of our knowledge, not yet been proposed.

Example 3.2. Let q be the Boolean conjunctive query $\{C(x), t(x, y), C(y)\}$ and $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ a $\mathcal{SHIQ}$ knowledge base with $\mathcal{T} = \{\top \sqsubseteq C \sqcup \neg C, \top \sqsubseteq \exists t.\top\}$ for t a transitive role, $\mathcal{R} = \emptyset$, and $\mathcal{A} = \{(\neg C)(a)\}$. As in the previous example, we clearly have that $\mathcal{K} \not\models q$. The axiom $\top \sqsubseteq C \sqcup \neg C$ makes sure that, for each node of a completion graph, we choose either C or $\neg C$, which then allows for a purely syntactic mapping of the query into the completion graph. The left hand side of Figure 3.6 shows a possible (but simplified) completion graph $\mathbf{G}$ for $\mathcal{K}$. The grey underlying triangle shapes illustrate the two isomorphic trees used for blocking, and clearly there is no mapping for q into $\mathbf{G}$ since there is only one node that has C in its label. Please note that we used just trees of depth one here since the longest path in q is of length one. The example can, however, equally be repeated with *any* desired depth for the blocking trees. The partial representation of a model depicted in the right hand side of Figure 3.6 shows that, by unravelling $\mathbf{G}$, we would get a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I}$ satisfies q. This is the case because there is no longer only one element in the extension of C, and all role relationships for q are satisfied since t is transitive. Please note that implicit transitive relationships are only shown in the figure where they are relevant for the query match. Even choosing a larger depth for the two trees would allow this to happen since the decision for C or $\neg C$ is purely non-deterministic.

The absence of a mapping for the query into the completion graph does,

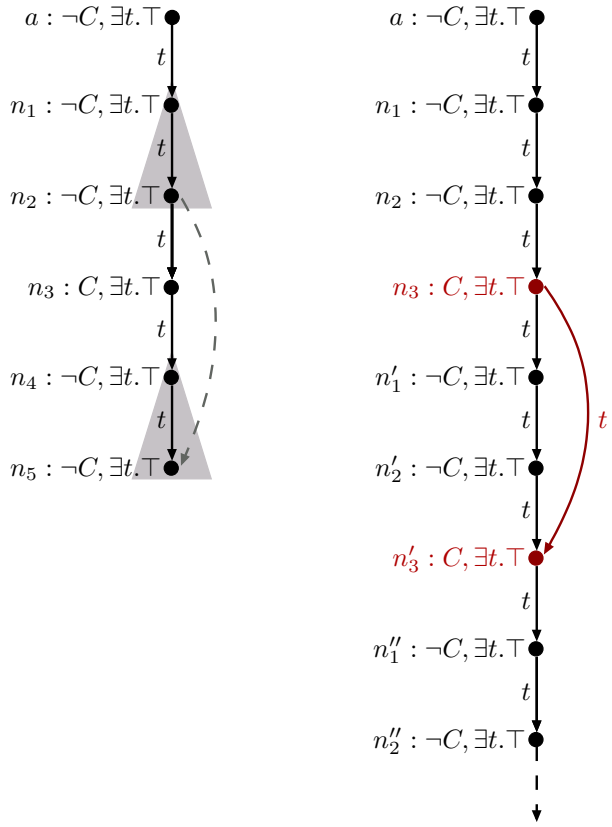

Figure 3.6: A completion graph **G** with trees for blocking (left) and a partial representation of a model $\mathcal{I}$ for $\mathcal{K}$ and q (right). The match for the query in $\mathcal{I}$ is shown in red.

therefore, not guarantee that the canonical model for the completion graph does not satisfy the query. Requiring several isomorphic trees or pairs before blocking occurs obviously does not help either since only one path between the isomorphic trees/pairs could contain the twice required C in the node label. The problem is that, due to the transitive edges that are only made explicit in the model, elements that are several steps away from each other can be directly related in the canonical model. By repeating the unique part between the blocking and the blocked node, we can thus produce a structure that suddenly provides a query match.

An extension of the CARIN-style algorithms to $\mathcal{SHOIQ}$, even when only simple roles are present in the query, is problematic for different reasons. Without nominals, a completion graph can be seen as a forest, where each individual from the ABox is the root of a tree. We can have arbitrary edges as induced by the role assertions in the ABox only between the roots. If we further add nominals

to the logic, this forest structure is blurred even further. Concepts of the form $\exists r.\{o\}$ with o a nominal, or similar concepts using atleast number restrictions, can now cause links from within a tree back to the node that represents the nominal o. This clearly has a consequence on the blocking conditions. If we want to guarantee that a match for the query in a canonical model can be "shifted up" as described in the proof sketch above, then a tree should also consider links back to nominals and all further edges up to a depth that depends on the query. For $\mathcal{SHOIQ}$, however, this spoils the termination proof since the algorithm generates also new nominal nodes in certain circumstances. These new nominal nodes should not be considered in the blocking condition since the bound on their number depends on the length of the longest path before blocking occurs. If the blocking condition takes these nodes into account, as required for the completeness proof as sketched above, then we have cyclic dependencies between the completeness and the termination proof and we have the choice of either being incomplete or non-terminating. Currently it is, to the best of our knowledge, not known if the bound on the number of new nominals generated by the tableau algorithm for $\mathcal{SHOIQ}$ [63] and the blocking conditions can be fixed independently of each other. In the conclusions, we comment further on the problems that arise when we try to extend query entailment algorithms from $\mathcal{SHIQ}$ or $\mathcal{SHOQ}$ to $\mathcal{SHOIQ}$.

3.6 Hybrid Logics

In this section, we introduce an algorithm for query entailment in $\mathcal{SHQ}$. The algorithm is only a decision procedure for a restricted class of conjunctive queries and $\mathcal{SHQ}$ is subsumed by both $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ for which we present decision procedures in the following two chapters. We nevertheless include the algorithm since it is based on a completely different technique that uses the $\downarrow$ operator known from Hybrid Logics [13]. We discuss the relationship of this technique with existing ones at the end of this section and illustrate the difficulties of extending it to more expressive logics.

3.6.1 Hybrid Logic Binders for Query Answering

An extension of the rolling-up technique to cyclic queries is not directly possible (cf. Chapter 1). Due to the tree model property of most DLs [54], a concept cannot

capture cyclic relationships, but it is also not always correct to replace variable names in a cycle with individual names from the knowledge base (cf. Section 3.4). In this section, we show how cyclic queries for an $\mathcal{SHQ}$ knowledge base can be rolled-up into $\mathcal{SHQ}_{\downarrow}$-concepts. The $\downarrow$ binder can label elements in a model with a variable, and this variable can be used to enforce a co-reference. For example, consider the simple cyclic query $q = \{r(y, y)\}$, which cannot straightforwardly be expressed as a Description Logic concept. With $\downarrow$ binders and variables, however, we can express the query by the following concept: $C_q = \downarrow y.(\exists r.y)$. A knowledge base $\mathcal{K}$ does not entail q iff $\mathcal{K}$ extended with the axiom $\top \sqsubseteq \neg C_q = \downarrow y.(\forall r.(\neg y))$ is consistent. A model for such an extended knowledge base is a model in which each element d is not an r-successor of itself because in such a model we can bind the variable y to each element in the domain and all reachable r-successors are different from y. More precisely, we define the syntax and semantics of the $\downarrow$ binder and concepts including variables as follows:

Definition 3.3. Let N_C, N_R, N_I, and N_V be countably infinite and pairwise disjoint sets of concept, role, individual, and variable names respectively, and let $\mathcal{L}$ be a Description Logic. With $\mathcal{L}_{\downarrow}$, we denote the language obtained by, additionally, allowing for y and $\downarrow y.C$ as concepts, for $y \in N_V$ and C an $\mathcal{L}_{\downarrow}$-concept over the signature $\mathcal{S} = (N_C, N_R, N_I, N_V)$.

For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, an element $d \in \Delta^{\mathcal{I}}$, and a variable $y \in N_V$, we denote with $\mathcal{I}_{[y/d]}$ the interpretation that extends $\mathcal{I}$ such that $y^{\mathcal{I}} = \{d\}$. The $\mathcal{L}_{\downarrow}$-concept $\downarrow y.C$ is then interpreted as $(\downarrow y.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid d \in C^{\mathcal{I}_{[y/d]}}\}$. △

We say that a variable y is *free* in an $\mathcal{L}_{\downarrow}$-concept C if y is not bound by $\downarrow$. We assume, without loss of generality, that every variable y is only bound once by an occurrence of $\downarrow y$. Every concept C in which this is not the case can be transformed into an equivalent one by naming variables apart [38].

For some queries, we also need role conjunctions and inverse roles in order to express the query as a concept. If the query contains constants or individual names, then nominals are very convenient for defining the rolling-up procedure.

Example 3.4. Let $q = \{s(x, x'), r_1(x, x'), s(x', a), r_2(x, y), r_3(y, z), r_4(z, x)\}$ be a Boolean conjunctive query with $\mathsf{Inds}(q) = \{a\}$ and $\mathsf{Vars}(q) = \{x', x, y, z\}$. Figure 3.7 shows a query graph for q. By using nominals, role conjunctions, and the $\downarrow$ binder, we can equivalently express the query as the concept (using x as the

starting node for the rolling-up):

$$C_q = \downarrow x.(\exists r_2.(\exists r_3.(\exists r_4.x))) \sqcap \exists(s \sqcap r_1).(\exists s.(\{a\}))$$

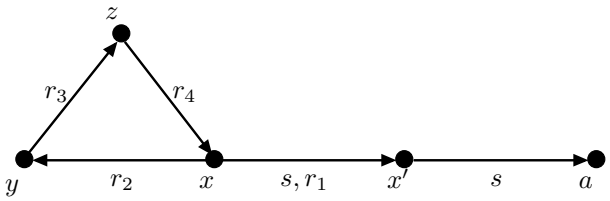

Figure 3.7: A graphical representation of the query q from Example 3.4.

Please note that, since we consider the DL $\mathcal{SHQ}$, the concept conjuncts of the query are limited to $\mathcal{SHQ}$-concepts and they contain, therefore, no nominals. Furthermore, the nominals that we introduce in the rolling-up, e.g., in the subconcept $\exists s.(\{a\})$ of C_q, always occur positively. In order to decide entailment of q, we decide whether the queried knowledge base extended with the axiom $\top \sqsubseteq \mathsf{nnf}(\neg C_q)$ is consistent, where $\mathsf{nnf}(\neg C_q)$ denotes the negation normal form of $\neg C_q$. Hence, nominals occur only negated in the extended knowledge base and cannot enforce any non-tree-shaped relational structures in a model of the extended knowledge base. It is, therefore, also possible to use representative concepts or name formulae [22, 129]. For this, we extend the original knowledge base with an assertion $N_a(a)$, for each individual name $a \in \mathsf{Inds}(\mathcal{A})$ and N_a a fresh concept name and use the concept N_a in the query instead of the nominal constructor $\{a\}$.

Adding inverse roles or role conjunction to $\mathcal{SHQ}$ is not a big problem, and we present a decision procedure for $\mathcal{SHIQ}^{\sqcap}$ knowledge base consistency in Chapter 4. Adding $\downarrow$ binders, however, makes even the relatively simple DL $\mathcal{ALC}$ undecidable [13]. We can observe, however, some interesting properties for our query concepts:

1. After the rolling-up, all variables occur only positively.

2. Only existential restrictions are introduced in the rolling-up.

Hence, by negating the query concept, as required for building an extended knowledge base, and by transforming it into negation normal form, we obtain a concept in which all variables occur negated and are under the scope of universal quantifiers. As a consequence, some kind of tree model property is still preserved.

For arbitrary concepts with variables and $\downarrow$ binders, this is clearly not the case. Adding an axiom $\top \sqsubseteq \downarrow y.(\exists r.y)$, for example, makes r a reflexive relation in each model of the knowledge base. Even more complicated concepts can be used that enforce models which do not have any form of tree model property.

An interesting consequence of extending the knowledge base with the negated query concepts is that we still loose the finite model property, which $\mathcal{SHQ}$ enjoys (i.e., if an $\mathcal{SHQ}$ knowledge base is satisfiable, then it is satisfiable in a model with finite domain). For example, let $q = \{t(y, y)\}$, which is rolled-up into the concept $\downarrow y.(\exists t.y)$. An extended knowledge base would then contain the axiom $\top \sqsubseteq \downarrow y.(\forall t.(\neg y))$. Assume further that the knowledge base contains also the axiom $\top \sqsubseteq \exists t.\top$ with t a transitive role. This axiom enforces a t-successor for every individual in the domain. Normally, a finite model could contain a t-cycle instead of an infinite chain, but this would clearly violate $\downarrow y.(\forall t.(\neg y))$ in case t is transitive. Hence, every model of $\mathcal{K}$ must be acyclic and, therefore, contain infinitely many individuals.

Since $\mathcal{SHQ}$ is a subset of both $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ and we present conjunctive query entailment decision procedures for both logics in the following two chapters, we mainly want to give an idea of this technique. For simplicity, we assume, therefore, that no inverse roles are required to build the query concept. We informally discuss later what is required when inverse roles are necessary to build the query concept.

Our algorithm is based on the tableau algorithm for $\mathcal{SHIQ}$ [68], which also decides knowledge base satisfiability for $\mathcal{SHQ}$. A rough sketch of a tableau algorithm and the required terminology is given in Section 3.5.1. We assume that representative concepts/name formulae [34, 129] are used instead of nominals for the constants in the query.

In order to handle query concepts that may contain binders and variables, some adaptations of the existing algorithm are necessary. In order to store the bindings of the variables, we modify the node labels such that they contain tuples of the form (C, B), where C is a concept and B is a (possibly empty) set of bindings of the form $\{y_1/v_1, \ldots, y_n/v_n\}$, where $y_1, \ldots, y_n$ are the free variables in C and $v_1, \ldots, v_n$ are nodes of the completion graph. The next obvious addition is a rule to handle concepts of the form $\downarrow y.C$: if $(\downarrow y.C, B)$ is in the label of a node v, we add $(C, \{y/v\} \cup B)$ and $(y, \{y/v\})$ to the label of v. The latter states that y is bound to v at the node v, and the former that the free variable y in C

is bound to v. All other existing rules have to propagate the bindings as well, e.g., the $\forall$-rule applied to $(\forall r.C, B)$ in the label of a node v adds (C, B) to the labels of v's r-successors. The set B contains all and only the bindings for the free variables in C. Another obvious consequence is the addition of a new clash condition: if both $(y, \{y/v\})$ and $(\neg y, \{y/v\})$ are in the label of the node v, then this is a clash.

A more challenging task is the extension of the blocking condition. In the absence of inverse roles, however, we argue that we can simply ignore the bindings, i.e., if (C, B) is in the label, we consider only C in the blocking condition. This clearly ensures termination. But why does this guarantee that we can unravel a complete and clash-free completion graph into a model? Obviously, without inverse roles, there is no way to propagate information "back" from a node to its ancestors and clashes according to the new clash condition can only occur through a cyclic structure. More precisely, a node v is only labelled with $(y, \{y/v\})$ by an application of the new $\downarrow$-rule to some concept $(\downarrow y.C, B)$ in the label of v. Furthermore, $\neg y$ with v as the binding for y can only occur in the label of v when $(C, \{y/v\} \cup B)$ is expanded to $(\neg y, \{y/v\})$ via a cyclic path back to v. Without inverse roles in the query concept, this is obviously only possible among individual nodes and, therefore, no clash in the tableau can by caused by unravelling.

The above arguments for ignoring the bindings in the blocking condition are obviously no longer valid when the query concept contains inverse roles. In this case, we have to consider also the bindings in the blocking condition. If we say, as usual, that a node is blocked if it has an ancestor that has the same label (more precisely, we have to consider pairs of nodes in the presence of inverse roles and number restrictions), then termination is no longer guaranteed because the bindings can be different for each node. We need, therefore, a weaker condition that is still strong enough to guarantee that a complete and clash-free completion graph can be unravelled into a model.

Please note that the bindings that we store together with the concepts make it possible to trace the expansion of concepts that contain free variables. We can use these traces in order to define a revised blocking condition by requiring that, for every pair (C, B) in the label of the blocked node, there is a pair (C, B') in the blocking node such that the trace for (C, B) is "isomorphic" to the trace of (C, B'), for a suitable definition of isomorphic in this context. If the query contains only simple roles, this form of blocking is a weak form of the

tree blocking that is used in the CARIN-style algorithms. It is weaker since we do not require whole trees, but only isomorphic traces. In particular, if the labels do not contain any bindings, we can use the normal blocking conditions, whereas in the CARIN-style algorithms tree blocking is always necessary. If the query contains also non-simple roles, an extension of the blocking conditions is not as straightforward. Consider again a knowledge base containing the axiom $\top \sqsubseteq \exists t.\top$ with t a transitive role and the query concept $\downarrow y.(\exists t.y)$, for which we extend the queried knowledge base with the axiom $\top \sqsubseteq \downarrow y.(\forall t.(\neg y))$. In this case, the expansion rules generate a new t-successor for each node due to the axiom $\top \sqsubseteq \exists t.\top$ and blocking is necessary to ensure termination. The axiom $\top \sqsubseteq \downarrow y.(\forall t.(\neg y))$, however, causes $(\forall t.(\neg y), \{y/v\})$ to be present in the label of each node v and, since t is transitive, the expansion rules add not only $(\neg y, \{y/v\})$ to v's t-successor, but also $(\forall t.(\neg y), \{y/v\})$. Hence, concepts of the form $\forall t.(\neg y)$ with different bindings for y accumulate the more we expand the completion tree and we cannot find isomorphic traces as required for blocking. A possible weakening of the blocking conditions could be the following: for each pair (C, B) in the blocked node, there is either a pair (C, B') in the blocking node such that the trace of (C, B') is isomorphic to the trace of (C, B), or (C, B) occurs also in the label of the blocking node. In this case, the concepts in the blocking node witness the further expansion of the concepts from the blocked node during the unravelling and provide us with enough information to know that no clash occurs in the unravelled tableau/model.

Summing up, an extension of this technique into a decision procedure for arbitrary conjunctive queries is non-trivial. We sketch a possible solution above and we conjecture that a further extension to the DL $\mathcal{SHIQ}$ would also be possible. We do not provide formal proofs here since we present a different decision procedure for $\mathcal{SHIQ}$ in the next chapter that is worst-case optimal regarding both data and combined complexity. It is left for future work to specify the details of the above sketched algorithm and prove its correctness.

3.7 First-Order Logic

Finally, another interesting question is whether or not there are decision procedures for fragments of First-Order Logic (FOL) that we can exploit for the purpose of deciding query entailment. A Boolean conjunctive query is an existential,

conjunctive FOL formula and it is well known that a $\mathcal{SHOIQ}$ knowledge base can be translated into $\mathcal{C}_2$, the two-variable fragment of FOL with counting quantifiers [132]. The satisfiability problem for $\mathcal{C}_2$ is decidable in NExpTime [100]. Hence, we can see the problem of deciding conjunctive query entailment with respect to a DL knowledge base also as deciding logical implication of a formula with respect to an FOL theory. To the best of our knowledge, there is no straightforward reduction of a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOIQ}$ knowledge base to a decidable fragment of FOL that can additionally capture conjunctive queries. One reason for this is that a query can contain an unbounded number of variables and, hence, does not belong to a bounded variable fragment as standard DLs do. In particular in the case of cyclic queries, we cannot eliminate variables in a straightforward way, e.g., by rewriting the query, since the variables are necessary to keep the co-reference in the cycle.

For expressive DLs such as the ones we consider here, conjunctive queries do also not directly correspond to the guarded fragment (GF) [4, 55] or any of its extensions. Formulae of the guarded fragment are such that all quantification is guarded, i.e., formulae of the guarded fragment have one of the following two forms:

$$\forall \vec{x}.\,(G \rightarrow F) \qquad \text{or} \qquad \exists \vec{x}.\,(G \wedge F)\,,$$

where G is an atom that contains all free variables of F. The atom G is called a *guard* for F. Simpler Description and Modal Logics naturally belong to the guarded fragment and, for many constructors of expressive DLs, e.g., nominals or number restrictions, decidable extensions are known [77]. One of the main problems, however, are transitive roles since an FOL formula that states the transitivity of a role is not guarded. E.g., an FOL formula to state that t is a transitive role is the following:

$$\forall x, y, z.\,(t(x, y) \wedge t(y, z) \rightarrow t(x, z))\,.$$

Please note that none of the two atoms $t(x, y)$ and $t(y, z)$, which could act as atom guards, contain all free variables from $t(x, z)$. For DL knowledge bases, this is usually not a problem since, even for $\mathcal{SHOIQ}$, checking the consistency of a knowledge base in a logic with transitive roles can polynomially be reduced to checking an extended knowledge base without transitive roles [74, 78, 91]. This

reduction can, however, not directly be applied to queries.[1]

In the guarded fragment with transitivity, one can, however, also directly declare some relations as transitive. In general, this fragment is undecidable even for the two-variable fragment of GF without equality [40], but when the transitive relations are only used in the guards, the fragment is decidable [128]. For conjunctive queries, the latter fragment is not sufficient since in conjunctive queries guards are not necessarily atomic since we can have queries that contain a conjunction of role atoms such as $t(x, y) \wedge t(y, z)$ for a transitive relation t. Such conjunctive queries are better captured by the *loosely guarded fragment* (LGF) [135] in which the notion of guards is relaxed. A guard may be a conjunction of atoms, which is naturally the case for conjunctive queries. Conjunctive queries with transitive roles belong, however, to LGF with transitive guards [$LGF+TG$], which is, again, undecidable even if only one transitive relation is used [80].

3.8 Summary

In this chapter we have introduced related areas and previous work. We introduced the first conjunctive query answering algorithm [22], which provides the ideas that are also part of the work presented in the following two chapters. We also show pitfalls that arise in the construction of query answering algorithms for very expressive logics and highlight sources of incompleteness in existing algorithms. We further point out connections to other areas in logic and show how the $\downarrow$ operator from Hybrid Logics can be used in the design of query answering algorithms. A generalisation of the suggested techniques to expressive logics is, however, a non-trivial task since the $\downarrow$ operator requires careful attention and, without restrictions, already makes the simple DL $\mathcal{ALC}$ undecidable. Finally, we briefly introduce the guarded and loosely guarded fragment, which is closely related to Description and Modal Logics. To the best of our knowledge, none of

[1] The standard encoding works by enforcing the propagation of concepts that are universally quantified along the transitive edges in the constructed model so that, by transitively closing the relations that are transitive in the original knowledge base, the universal restrictions are still not violated. Checking query entailment on the constructed model (without the transitively closed relations) is, however, not always possible since queries can also require checking the relational structures of a model. When we find no match in the constructed model and conclude that the query entailment does not hold, we might make the wrong decision since after extending the relations that are transitive in the original knowledge base, the query could be true.

its decidable extensions can directly express conjunctive queries. Hence, a solution for the decision problem of conjunctive query entailment cannot be found by a simple translation into known decidable fragments of FOL although more involved translations might exist. Summing up, this chapter contains rather discouraging observations regarding the decidability problem for conjunctive queries in expressive DLs. In the next two chapters we show, however, that for the DLs $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ this problem is decidable, although of a higher complexity than the standard reasoning problems.

Chapter 4

Query Entailment for $\mathcal{SHIQ}$

In this chapter, we present a decision procedure for answering unions of conjunctive queries (UCQs) with respect to $\mathcal{SHIQ}$ knowledge bases. $\mathcal{SHIQ}$ is obtained from $\mathcal{SHOIQ}$ by disallowing nominals. More precisely, for $\mathcal{S} = (N_C, N_R, N_I)$ a signature, the set of $\mathcal{SHIQ}$-*concepts* is the smallest set built inductively over $\mathcal{S}$ using the following grammar, where $A \in N_C, n \in \mathbb{N}_0, r$ is a role, and s is a simple role:

$$C ::= \top \mid \bot \mid A \mid \neg A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall r.C \mid \exists r.C \mid \leqslant n\ s.C \mid \geqslant n\ s.C.$$

Please note that the above definition ensures that all $\mathcal{SHIQ}$-concepts are automatically in negation normal form.

In our query entailment decision procedure we allow, in particular, for non-distinguished variables and transitive (more precisely, non-simple roles) in the query body. Previously existing algorithms forbid either one of these two features: without non-distinguished variables, query answering trivially reduces to instance retrieval; with the restriction to simple roles in the query body, the paths in the completion graph that can be used in the query mapping are of bounded length (cf. Section 3.5.1).

When we use the term conjunctive query in this section, we implicitly mean $\mathcal{SHIQ}$ conjunctive query, i.e., the concepts in concepts conjuncts are $\mathcal{SHIQ}$-concepts, and the roles in role conjuncts are possibly inverse roles.

In the algorithm for $\mathcal{SHIQ}$, we rewrite a conjunctive query into a set of conjunctive queries such that each resulting query is either tree-shaped (i.e., it can be expressed as a concept) or grounded (i.e., it contains only constants/individual names and no variables). For both types of queries, entailment can be reduced

to standard reasoning tasks [22, 64]. The grounded queries contain also the parts of the input query that were already grounded because of individual names that occurred in concept or role conjuncts (i.e., when $\mathsf{Inds}(q) \neq \emptyset$). In Section 4.1, we motivate the query rewriting steps by means of an example. In Section 4.2, we give exact definitions for the rewriting procedure and show that a Boolean query is indeed entailed by a knowledge base $\mathcal{K}$ iff the disjunction of the rewritten queries is entailed by $\mathcal{K}$. In Section 4.3, we present a deterministic algorithm for UCQ entailment in $\mathcal{SHIQ}$ that runs in time single exponential in the size of the knowledge base and double exponential in the size of the query. We further show that conjunctive query entailment in $\mathcal{SHIQ}$ is co-NP-complete regarding data complexity, and thus not harder than instance retrieval.

4.1 Query Rewriting by Example

In this section, we motivate the ideas behind our query rewriting technique and introduce the different rewriting steps by means of an example. In the following section, we give precise definitions for all rewriting steps.

4.1.1 Forest Bases and Canonical Interpretations

Similarly to the technique introduced in the CARIN system, the main observation is that we can focus on the canonical models of the knowledge base, i.e., models that have a kind of tree or forest shape. In the CARIN system, completion trees are used as finite representations of the canonical models, and a Boolean query is entailed by the knowledge base iff the query can be mapped to every complete and clash-free completion tree. Here we proceed in the opposite way: we use the rewritten queries in order to construct a canonical model in which the query is false.

Intuitively, a tree model is an interpretation whose domain consists of a tree in which the successors of a node n are those elements that are related to n via some role. It is well known that every satisfiable $\mathcal{ALC}$-concept does have interpretations that have such a tree shape [53, 140]. For satisfiable or consistent knowledge bases, this property generalises to a forest model property, where the domain of an interpretation consists of a collection of trees such that each tree is rooted in an element that corresponds to an individual named in the knowledge base. We call these forest shaped models canonical models. If the logic allows for

transitive roles, and $\mathcal{SHIQ}$ clearly does, we cannot restrict tree or forest models to those interpretations where a node n is related to its r-successor for some role r. For example, if t is a transitive roles, an element n that has a t-successor n' that itself has a t-successor n'', must have n'' as a successor for n.

In order to still define models whose domain consists of a collection of trees, we also introduce *forest bases*, which are forest-shaped interpretations that interpret transitive roles in an unrestricted way, i.e., not necessarily in a transitive way. In particular, we require that a forest base does not include *any* relationships between elements of the domain that could be inferred due to role transitivity. The term tree, as used below, is to be understood as in Definition 2.7 on page 39. In the following, we assume that the ABox contains at least one individual name, i.e., $\mathsf{Inds}(\mathcal{A})$ is non-empty. This is without loss of generality since we can always add an assertion $\top(a)$ to the ABox for a fresh individual name $a \in N_I$.

For readers familiar with tableau algorithms, it is worth noting that forest bases can also be thought of as those tableaux generated from a complete and clash-free completion graph [68].

We now define more precisely what we mean by canonical or forest models and we show that, for deciding query entailment, we can restrict our attention to forest models. We then explain how the rewriting steps are used to transform cyclic subparts of the query into tree-shaped ones such that the resulting query can be mapped to a forest model.

Definition 4.1. A *forest base for* $\mathcal{K}$ is an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ that interprets transitive roles in an unrestricted (i.e., not necessarily transitive) way and, additionally, satisfies the following conditions:

- T1 $\Delta^{\mathcal{J}} \subseteq \mathsf{Inds}(\mathcal{A}) \times \mathbb{N}^*$ such that, for all $a \in \mathsf{Inds}(\mathcal{A})$, the set $\{w \mid (a, w) \in \Delta^{\mathcal{J}}\}$ is a tree;
- T2 if $((a, w), (a', w')) \in r^{\mathcal{J}}$, then either $w = w' = \varepsilon$ or $a = a'$ and w' is a neighbour of w; and
- T3 for each $a \in \mathsf{Inds}(\mathcal{A})$, $a^{\mathcal{J}} = (a, \varepsilon)$.

An interpretation $\mathcal{I}$ is *canonical for* $\mathcal{K}$ if there exists a forest base $\mathcal{J}$ for $\mathcal{K}$ such that $\mathcal{I}$ is identical to $\mathcal{J}$ except that, for all non-simple roles r, we have

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq^*_{\mathcal{R}} r,\ s \in \mathsf{Trans}_{\mathcal{R}}} (s^{\mathcal{J}})^+.$$

In this case, we say that $\mathcal{J}$ is a forest base *for* $\mathcal{I}$ and, if $\mathcal{I} \models \mathcal{K}$, we say that $\mathcal{I}$ is a *canonical model for* $\mathcal{K}$. △

For convenience, we extend the notion of successors and neighbours to elements in canonical models. Let $\mathcal{I}$ be a canonical model with $(a, w), (a', w') \in \Delta^{\mathcal{I}}$. We call (a', w') a *successor* of (a, w) if either $a = a'$ and $w' = w \cdot c$ for some $c \in \mathbb{N}$ or $w = w' = \varepsilon$. We call (a', w') a *neighbour* of (a, w) if (a', w') is a successor of (a, w) or vice versa.

Please note that the above definition implicitly relies on the unique name assumption (UNA) (cf. T3). Making the UNA simplifies the above and some of the following definitions. In Section 4.3, we show that this assumption is without loss of generality and we show how our decision procedure can be extended to the case where we do not make the UNA and that the complexity results carry over.

Lemma 4.2. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and q a union of conjunctive queries, then $\mathcal{K} \not\models q$ iff there exists a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

For the only if direction of the proof, we take an arbitrary counter-model for the query, which exists by assumption, and "unravel" all non-tree structures. Since, during the unraveling process, we only replace cycles in the model with infinite paths and leave the interpretation of concepts unchanged, the query is still not satisfied in the unravelled canonical model.

Proof. The "if" direction is trivial.

For the "only if" direction, since an inconsistent knowledge base entails every query, we can assume that $\mathcal{K}$ is consistent. Hence, there is an interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ such that $\mathcal{I}' \models \mathcal{K}$ and $\mathcal{I}' \not\models q$. From $\mathcal{I}'$, we construct a canonical model $\mathcal{I}$ for $\mathcal{K}$ and its forest base $\mathcal{J}$ as follows: we define the set $P \subseteq (\Delta^{\mathcal{I}'})^*$ of *paths* to be the smallest set such that

- for all $a \in \mathsf{Inds}(\mathcal{A}), a^{\mathcal{I}'}$ is a path;
- $d_1 \cdots d_n \cdot d$ is a path, if
  - $d_1 \cdots d_n$ is a path,
  - $(d_n, d) \in r^{\mathcal{I}'}$ for some role r,
  - if $n > 1$, then there is no $a \in \mathsf{Inds}(\mathcal{A})$ such that $d_2 = a^{\mathcal{I}'}$,

– if $n > 2$, then $d \neq d_{n-1}$.

Please note that the second to last condition is to prevent the unravelling cycles within the ABox and the last condition is necessary in the presence of inverse roles and ensures the proper handling of links to the predecessors and correct unravelling of loops.

Now fix a set $S \subseteq \mathsf{Inds}(\mathcal{A}) \times \mathbb{N}^*$ and a bijection $f\colon S \to P$ such that, for each $a \in \mathsf{Inds}(\mathcal{A})$,

(i) $(a, \varepsilon) \in S$,

(ii) $\{w \mid (a, w) \in S\}$ is a tree,

(iii) $f(a, \varepsilon) = a^{\mathcal{I}'}$, and

(iv) if $(a, w), (a, w') \in S$ with w' a successor of w, then $f(a, w') = f(a, w) \cdot d$ for some $d \in \Delta^{\mathcal{I}'}$.

For all $(a, w) \in S$, set $\mathsf{Tail}(a, w) = d_n$ if $f(a, w) = d_1 \cdots d_n$. We now define a forest base $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ for $\mathcal{K}$ as follows:

(a) $\Delta^{\mathcal{J}} = S$;

(b) for each $a \in \mathsf{Inds}(\mathcal{A})$, $a^{\mathcal{J}} = (a, \varepsilon) \in S$;

(c) for each $A \in N_C$ and $(a, w) \in S$, $(a, w) \in A^{\mathcal{J}}$ iff $\mathsf{Tail}(a, w) \in A^{\mathcal{I}'}$;

(d) For all roles r, $((a, w), (b, w')) \in r^{\mathcal{J}}$ if either

(I) $w = w' = \varepsilon$ and $(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in r^{\mathcal{I}'}$ or

(II) $a = b$, w' is a neighbour of w, and $(\mathsf{Tail}(a, w), \mathsf{Tail}(b, w')) \in r^{\mathcal{I}'}$.

It is clear that $\mathcal{J}$ is a forest base for $\mathcal{K}$ due to the definition of S and the construction of $\mathcal{J}$ from S.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that is identical to $\mathcal{J}$ except that, for all non-simple roles r, we set

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq^*_{\mathcal{R}} r,\ s \in \mathsf{Trans}_{\mathcal{R}}} (s^{\mathcal{J}})^+.$$

We demonstrate why $\mathcal{I}$ is indeed a model of $\mathcal{K}$ by showing that, for each $(a, w) \in \Delta^{\mathcal{I}}$ and concept $C \in N_C$, $(a, w) \in C^{\mathcal{I}}$ iff $\mathsf{Tail}(a, w) \in C^{\mathcal{I}'}$.

We start with the if direction and prove this inductively on the structure of concepts: for the base case of atomic concepts and their negation, i.e., $C = A$ or $C = \neg A$ for $A \in N_C$, this clearly follows from (c). The cases for disjunction and conjunction, i.e., when $C = C_1 \sqcup C_2$ and $C = C_1 \sqcap C_2$ are straightforward by induction and since $\mathcal{I}' \models \mathcal{K}$.

Let $C = (\exists r.D)$ and $(a, w) \in C^{\mathcal{I}}$. Hence, there is an element (b, w') such that $((a, w), (b, w')) \in r^{\mathcal{I}}$ and $(b, w') \in D^{\mathcal{I}}$. By definition of the forest base $\mathcal{J}$, there is a sequence of elements $d_1, \ldots, d_n$ such that $d_1 = (a, w)$, $d_n = (b, w')$, and, for each i with $1 \leq i < n$, $(d_i, d_{i+1}) \in r^{\mathcal{J}}$. By (d) we additionally have that, for each i with $1 \leq i < n$, the pair (d_i, d_{i+1}) with $d_i = (a_i, w_i)$ and $d_{i+1} = (a_{i+1}, w_{i+1})$ is such that either

1. $w_i = w_{i+1} = \varepsilon$ and $(a_i^{\mathcal{I}'}, a_{i+1}^{\mathcal{I}'}) = (\mathsf{Tail}(a_1, w_1), \mathsf{Tail}(a_{i+1}, w_{i+1})) \in r^{\mathcal{I}'}$ or
2. $a_i = a_{i+1}$, w_{i+1} is a neighbour of w_i and $(\mathsf{Tail}(a_i, w_i), \mathsf{Tail}(a_{i+1}, w_{i+1})) \in r^{\mathcal{I}'}$.

By definition of the semantics, we then have that $(\mathsf{Tail}(a, w), \mathsf{Tail}(b, w')) \in r^{\mathcal{I}'}$ and, by induction, that $\mathsf{Tail}(b, w') \in D^{\mathcal{I}'}$, which proves the claim.

Let $C = (\forall r.D)$ and $(a, w) \in C^{\mathcal{I}}$. Assume, to the contrary of what is to be shown that $d = \mathsf{Tail}(a, w) \notin C^{\mathcal{I}'}$. Hence, there is an element d' such that $(d, d') \in r^{\mathcal{I}'}$ and $d' \in (\neg D)^{\mathcal{I}'}$. By construction of $\mathcal{I}$, we have that $f(a, w) = d_1 \cdots d_k$ with $d_k = d$ and $d_1 \cdots d_k$ is a path. We distinguish two cases:

1. The path $d_1 \cdots d_k$ is of length one, i.e., $k = 1$. Hence, there is an individual $a \in \mathsf{Inds}(\mathcal{A})$ such that $a^{\mathcal{I}'} = d$ and $w = \varepsilon$. We now again distinguish two cases:
   (a) There is an individual $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d'$. Then, d' is a path and, by construction of $\mathcal{I}$, $(a, \varepsilon), (b, \varepsilon) \in \Delta^{\mathcal{I}}$ and $((a, \varepsilon), (b, \varepsilon)) \in r^{\mathcal{I}}$. By induction, $(b, \varepsilon) \in (\neg D)^{\mathcal{I}}$, contradicting the initial assumption that $(a, \varepsilon) \in (\forall r.D)^{\mathcal{I}}$.
   (b) There is no individual $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d'$. Then, $d \cdot d'$ is a path and, by construction of $\mathcal{I}$, $d = (a, \varepsilon)$ and $d' = (a, c)$ for some $c \in \mathbb{N}$. By (d), $((a, \varepsilon), (a, c)) \in r^{\mathcal{I}}$ and, by induction, $(a, c) \in (\neg D)^{\mathcal{I}}$, contradicting the initial assumption that $(a, \varepsilon) \in (\forall r.D)^{\mathcal{I}}$.
2. The path $d_1 \cdots d_k$ is of length greater than one. Since $(d, d') \in r^{\mathcal{I}'}$ by assumption, $d_1 \cdots d_k \cdot d'$ is a path and, by construction of $\mathcal{I}$, $(a, w \cdot c) \in \Delta^{\mathcal{I}}$

for some $c \in \mathbb{N}$ and, by (d), $((a, w), (a, w \cdot c)) \in r^{\mathcal{I}}$. Now, by induction, $(a, w \cdot c) \in (\neg D)^{\mathcal{I}}$, which again contradicts the initial assumption. This finishes the proof of the claim.

Let $C = (\geqslant n\ s.D)$ and $(a, w) \in C^{\mathcal{I}}$. Then (a, w) has n distinct s-successors $(a_1, w_1), \ldots, (a_n, w_n)$ such that, for each i with $1 \leq i \leq n$, either $w = w_i = \varepsilon$ or $a = a_i$ and w_i is a neighbour of w. In both cases, by construction of the set P and since f is a bijection, also $\mathsf{Tail}(a, w)$ has n distinct s-successors $d_1, \ldots, d_n$ such that, by induction, $d_i \in D^{\mathcal{I}'}$ for each i with $1 \leq i \leq n$. Hence, $\mathsf{Tail}(a, w) \in C^{\mathcal{I}'}$ as required.

Let $C = (\leqslant n\ s.D)$ and $(a, w) \in C^{\mathcal{I}}$. We assume, to the contrary of what is to be shown, that $\mathsf{Tail}(a, w) \in (\geqslant n+1\ s.D)^{\mathcal{I}'}$. Let $f(a, w) = d_1 \cdots d_k$, then $\mathsf{Tail}(a, w) = d_k$ has at least $n + 1$ distinct s-successors $d_s^1, \ldots, d_s^{n+1}$ such that, for each i with $1 \leq i \leq n + 1$, $d_s^i \in D^{\mathcal{I}'}$. We distinguish two cases:

1. The path $d_1 \cdots d_k$ is of length one, i.e., $k = 1$. Hence, $a^{\mathcal{I}'} = d_k$ and $w = \varepsilon$. For each d_s^i with $1 \leq i \leq n + 1$, we again distinguish two cases:

   (a) There is some $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$. Then, by definition of paths, $d_s^i \in P$ and, by construction of $\mathcal{J}$ and by (iii), $(b, \varepsilon) \in S$ with $f(b, \varepsilon) = b^{\mathcal{I}'}$. Then, by (d), $((a, \varepsilon), (b, \varepsilon)) \in s^{\mathcal{I}}$ and, by induction, $(b, \varepsilon) \in D^{\mathcal{I}}$.

   (b) There is no $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$. Hence, $d_1 \cdots d_k \cdot d_s^i$ is a path and, by construction of $\mathcal{I}$, $(a, w \cdot c) \in S$ for some $c \in \mathbb{N}$ and $f(a, w \cdot c) = d_1 \cdots d_k \cdot d_s^i$. By (d), $((a, w), (a, w \cdot c)) \in s^{\mathcal{I}}$ and, by induction, $(a, w \cdot c) \in D^{\mathcal{I}}$.

2. The path $d_1 \cdots d_k$ is of length greater than one. Then, by definition of paths, $d_1 \cdots d_k \cdot d_s^i$ is also a path. By construction of $\mathcal{I}$, $(a, w \cdot c) \in \Delta^{\mathcal{I}}$ for some $c \in \mathbb{N}$ and $f(a, w \cdot c) = d_1 \cdots d_k \cdot d_s^i$. By (d), $((a, w), (a, w \cdot c)) \in s^{\mathcal{I}}$ and, by induction, $(a, w \cdot c) \in D^{\mathcal{I}}$.

Since each d_s^i is distinct by assumption, we have that (a, w) has $n+1$ s-successors, which belong to the extension of D in $\mathcal{I}$, contradicting the initial assumption.

For the only if direction, we again prove the claim inductively on the structure of concepts. The base case of concept names and their negation again follows directly from the definition of $\mathcal{I}$, in particular from (c). The cases of disjunction and conjunction are also straightforward.

Let $C = (\exists r.D)$ and $\mathsf{Tail}(a, w) = d \in C^{\mathcal{I}'}$. Hence, there is an element $d' \in \Delta^{\mathcal{I}'}$ such that $(d, d') \in r^{\mathcal{I}'}$ and $d' \in D^{\mathcal{I}'}$. We distinguish two cases:

1. There is an individual name $a \in \mathsf{Inds}(\mathcal{A})$ such that $a^{\mathcal{I}'} = d$. By definition of $\mathcal{J}$ this implies that $w = \varepsilon$ and $f(a, w) = d$. We again distinguish two cases:

   (a) There is an individual name $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d'$. By definition of paths, d' is a path and, by definition of $\mathcal{J}$, $(b, \varepsilon) \in S$ with $f(b, \varepsilon) = d'$. By (d) and since $(d, d') \in r^{\mathcal{I}'}$, we have that $((a, \varepsilon), (b, \varepsilon)) \in r^{\mathcal{I}}$ and, by induction, that $(b, \varepsilon) \in D^{\mathcal{I}}$. Then, by definition of the semantics, $(a, w) = (a, \varepsilon) \in C^{\mathcal{I}}$ as required.

   (b) There is no individual name $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d'$. By definition of paths, $d \cdot d'$ is a path and, by definition of $\mathcal{J}$, there is some $c \in \mathbb{N}$ such that $f(a, w \cdot c) = d \cdot d'$. By (d), we then have that $((a, w), (a, w \cdot c)) \in r^{\mathcal{I}}$ and, by induction, $(a, w \cdot c) \in D^{\mathcal{I}}$. Hence, $(a, w) \in C^{\mathcal{I}} = (\exists r.D)^{\mathcal{I}}$ as required.

2. There is no individual name $a \in \mathsf{Inds}(\mathcal{A})$ such that $a^{\mathcal{I}'} = d$. Hence, $f(a, w) = d_1 \cdots d_k$ with $d_k = d$. By definition of paths and since $(d, d') \in r^{\mathcal{I}'}$, $d_1 \cdots d_k \cdot d'$ is a path and, by definition of S and $\mathcal{J}$, there is a $c \in \mathbb{N}$ such that $(a, w \cdot c) \in S$ and $f(a, \cdot c) = d_1 \cdots d_k \cdot d'$. By (d), we then have that $((a, w), (a, w \cdot c)) \in r^{\mathcal{I}}$ and, by induction, $(a, w \cdot c) \in D^{\mathcal{I}}$, which yields that $(a, w) \in C^{\mathcal{I}}$ as required.

Let $C = (\forall r.D)$ and let $\mathsf{Tail}(a, w) = d \in C^{\mathcal{I}'}$. Assume, to the contrary of what is to be shown, that $(a, w) \notin C^{\mathcal{I}}$, i.e., there is an element (a', w') such that $((a, w), (a', w')) \in r^{\mathcal{I}}$, and $(a', w') \in (\neg D)^{\mathcal{I}}$. We distinguish two cases:

1. The element (a', w') is a root, i.e., $w' = \varepsilon$. Then, by definition of S, there is an element $d' \in \Delta^{\mathcal{I}'}$ such that $a'^{\mathcal{I}'} = d'$ and $(d, d') \in r^{\mathcal{I}'}$. Since, by induction, $\mathsf{Tail}(a', w') = d' \in (\neg D)^{\mathcal{I}'}$, we have that $d \in (\exists r.(\neg D))^{\mathcal{I}'}$, which is a contradiction to our initial assumption.

2. The element (a', w') is not a root. Then, $a' = a$ and w' is a neighbour of w, i.e., either $w' = w \cdot c$ or $w = w' \cdot c$ for some $c \in \mathbb{N}$. By definition of f and by (d), there is some $d' \in \Delta^{\mathcal{I}'}$ such that $(d, d') \in r^{\mathcal{I}'}$ and, by induction, $d' \in (\neg D)^{\mathcal{I}'}$. Hence, $d \in (\exists r.(\neg D))^{\mathcal{I}'}$, which contradicts our initial assumption.

Let $C = (\geqslant n\ s.D)$ and let $\mathsf{Tail}(a, w) \in C^{\mathcal{I}'}$. By definition of the semantics, $\mathsf{Tail}(a, w)$ has at least n distinct s-successors $d_s^1, \ldots, d_s^n$ such that, for each i with $1 \leq i \leq n$, $d_s^i \in D^{\mathcal{I}'}$. Let $d_1 \cdots d_k$ be the path such that $f(a, w) = d_1 \cdots d_k$ and $d_k = \mathsf{Tail}(a, w)$. Such a path exists since f is a bijection and by definition of $\mathcal{I}$ and Tail. We distinguish three cases:

1. The path $d_1 \cdots d_k$ is of length one, i.e., $k = 1$. Hence, $a^{\mathcal{I}'} = d_k$ and $w = \varepsilon$. For each d_s^i with $1 \leq i \leq n$, we again distinguish two cases:

   (a) There is some $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$. Then, by definition of paths, $d_s^i \in P$ and, by construction of $\mathcal{J}$ and by (iii), $(b, \varepsilon) \in S$ with $f(b, \varepsilon) = b^{\mathcal{I}'}$. Then, by (d), $((a, \varepsilon), (b, \varepsilon)) \in s^{\mathcal{I}}$ and, by induction, $(b, \varepsilon) \in D^{\mathcal{I}}$.

   (b) There is no $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$. Hence, $d_1 \cdots d_k \cdot d_s^i$ is a path and, by construction of $\mathcal{I}$, $(a, c) \in S$ for some $c \in \mathbb{N}$ and $f(a, c) = d_1 \cdots d_k \cdot d_s^i$. By (d), $((a, \varepsilon), (a, c)) \in s^{\mathcal{I}}$ and, by induction, $(a, c) \in D^{\mathcal{I}}$.

2. The path $d_1 \cdots d_k$ is of length two, i.e., $k = 2$. By definition of paths, we have that $w \in \mathbb{N}$. For each d_s^i with $1 \leq i \leq n$, we again distinguish two cases:

   (a) If $d_s^i = d_1$, then $d_s^i = a^{\mathcal{I}'}$ and $(a, \varepsilon) \in S$ with $f(a, \varepsilon) = a^{\mathcal{I}'}$. By (d), $((a, c), (a, \varepsilon)) \in s^{\mathcal{I}}$ and, by induction, $(a, \varepsilon) \in D^{\mathcal{I}}$.

   (b) If $d_s^i \neq d_1$, the definition of paths implies that there is either some $b \neq a \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$ or there is no $b \in \mathsf{Inds}(\mathcal{A})$ such that $b^{\mathcal{I}'} = d_s^i$. In both cases, $d_1 \cdots d_k \cdot d_s^i$ is a path and, by construction of $\mathcal{I}$, $(a, w \cdot c) \in S$ for some $c \in \mathbb{N}$ and $f(a, w \cdot c) = d_1 \cdots d_k \cdot d_s^i$. By (d), $((a, w), (a, w \cdot c)) \in s^{\mathcal{I}}$ and, by induction, $(a, w \cdot c) \in D^{\mathcal{I}}$.

3. The path $d_1 \cdots d_k$ is of length greater than two. For each d_s^i with $1 \leq i \leq n$, we again distinguish two cases:

   (a) If $d_s^i = d_{k-1}$, then, by definition of paths, f, and S, $w = w' \cdot c$ with $w' \in \mathbb{N}^*, c \in \mathbb{N}$, and $f(a, w') = d_1 \cdots d_{k-1}$. By (d), $((a, w'), (a, w)) \in s^{\mathcal{I}}$ and, by induction, $(a, w') \in D^{\mathcal{I}}$.

(b) If $d_s^i \neq d_{k-1}$, then $d_1 \cdots d_k \cdot d_s^i$ is a path and there is some $c \in \mathbb{N}$ such that $f(a, w \cdot c) = d_1 \cdots d_k \cdot d_s^i$. Then, by (d), $((a,), (a, w \cdot c)) \in s^{\mathcal{I}}$ and $(a, w \cdot c) \in D^{\mathcal{I}}$ by induction.

Since each d_s^i is distinct by assumption, we have that (a, w) has n distinct s-successors, which belong to the extension of D in $\mathcal{I}$. Hence, $(a, w) \in C^{\mathcal{I}} = (\geqslant n\ s.D)^{\mathcal{I}}$.

Let $C = (\leqslant n\ s.D)$ and $\mathsf{Tail}(a, w) \in C^{\mathcal{I}'}$. We assume to the contrary of what is to be shown that $(a, w) \in (\geqslant n+1\ s.D)^{\mathcal{I}}$. Then (a, w) has at least $n+1$ distinct s-successors $(a_1, w_1), \ldots, (a_{n+1}, w_{n+1})$ such that, for each i with $1 \leq i \leq n+1, (a_i, w_i) \in D^{\mathcal{I}}$. We distinguish two cases:

1. The element (a, w) is a root, i.e., $w = \varepsilon$. By (iii), there is some $d \in \Delta^{\mathcal{I}'}$ such that $a^{\mathcal{I}'} = d$. We again distinguish two cases:

   (a) The element (a_i, w_i) is a root, i.e., $w_i = \varepsilon$. By (iii), there is some $d_s^i \in \Delta^{\mathcal{I}'}$ such that $a_i{}^{\mathcal{I}'} = d_s^i$. Since $((a, w), (a_i, w_i)) \in s^{\mathcal{I}}$ by assumption and by (d), we have that $(d, d_s^i) \in s^{\mathcal{I}'}$ and, by induction, that $d_s^i \in D^{\mathcal{I}'}$.

   (b) The element (a_i, w_i) is not a root. By definition of paths, since f is a bijection, by (ii), and (iv), we have that $a_i = a$ and $w_i = c$ for some $c \in \mathbb{N}$. Furthermore, $f(a, c) = d \cdot d_s^i$ for some $d_s^i \in \Delta^{\mathcal{I}'}$. By definition of paths and Tail, we then have that $(d, d_s^i) \in s^{\mathcal{I}'}$ and, by induction, that $d_s^i \in D^{\mathcal{I}'}$.

2. The element (a, w) is a not root, i.e., $w \neq \varepsilon$. Then, by (ii) and (iv), we have that $a_i = a$ and either $w_i = w \cdot c$ or $w = w_i \cdot c$ for some $c \in \mathbb{N}$. In the former case, we have by definition of $\mathcal{J}$ and paths, that $f(a, w) = d_1 \cdots d_k$ and $f(a_i, w_i) = d_1 \cdots d_k \cdot d_s^i$ for some $d_s^i \in \Delta^{\mathcal{I}'}$. By (d) we then have that $(d_k, d_s^i) \in s^{\mathcal{I}'}$ and $d_s^i \in D^{\mathcal{I}'}$ by induction. In the latter case, we have, by definition of $\mathcal{J}$ and paths, that $f(a, w) = d_1 \cdots d_{k-1} \cdot d_k$ and $f(a_i, w_i) = d_1 \cdots d_{k-1}$ with $d_{k-1} = d_s^i$. Again by (d), we then have that $(d_k, d_s^i) \in s^{\mathcal{I}'}$ and $d_s^i \in D^{\mathcal{I}'}$ by induction.

Since each d_s^i is distinct by assumption, this implies that $\mathsf{Tail}(a, w) \in (\geqslant n+1\ s.D)^{\mathcal{I}'}$, which contradicts the initial assumption.

Since $\mathcal{J}$ is a forest base, it follows that $\mathcal{I}$ is a canonical model for $\mathcal{K}$.

Therefore, we only have to show that $\mathcal{I} \not\models q$. Assume to the contrary that $\mathcal{I} \models q$ and let $q = q_1 \vee \ldots \vee q_n$. Then there is some π and i with $1 \leq i \leq n$

such that $\mathcal{I} \models^{\pi} q_i$. We now define a mapping $\pi' \colon \mathsf{Terms}(q_i) \to \Delta^{\mathcal{I}'}$ by setting $\pi'(t) = \mathsf{Tail}(\pi(t))$ for all $t \in \mathsf{Terms}(q_i)$. It is not difficult to check that $\mathcal{I}' \models^{\pi'} q_i$ and hence $\mathcal{I}' \models^{\pi'} q$, which is a contradiction. □

4.1.2 The Running Example

We use the following Boolean query and knowledge base as a running example:

Example 4.3. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $r, t \in N_{tR}, k \in \mathbb{N}$

$$
\begin{aligned}
\mathcal{T} = \ & \{ \ C_k \sqsubseteq \geqslant k\ p.\top, \\
& \quad C_3 \sqsubseteq \geqslant 3\ p.\top, \\
& \quad D_2 \sqsubseteq \exists s^-.\top \sqcap \exists t.\top \\
& \} \\
\mathcal{R} = \ & \{ \ t \sqsubseteq t^-, \\
& \quad s^- \sqsubseteq r \\
& \} \\
\mathcal{A} = \ & \{ \ r(a,b), \\
& \quad (\exists p.C_k \sqcap \exists p.C \sqcap \exists r^-.C_3)(a), \\
& \quad (\exists p.D_1 \sqcap \exists r.D_2)(b) \\
& \}
\end{aligned}
$$

and $q = \{r(u,x), r(x,y), t(y,y), s(z,y), r(u,z)\}$ with $\mathsf{Inds}(q) = \emptyset$ and $\mathsf{Vars}(q) = \{u, x, y, z\}$.

For simplicity, we choose to use a CQ instead of a UCQ. In the case of a UCQ, the rewriting steps are applied to each disjunct separately.

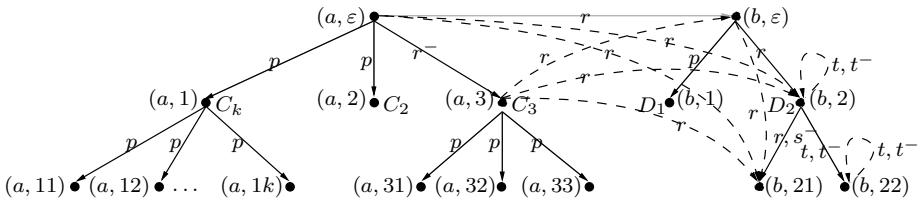

Figure 4.1: A representation of a canonical interpretation $\mathcal{I}$ for $\mathcal{K}$.

Figure 4.1 shows a representation of a canonical model $\mathcal{I}$ for the knowledge base $\mathcal{K}$ from Example 4.3. Each labelled node represents an element in the domain, e.g., the individual name a is represented by the node labelled (a, ε). The

edges represent relationships between individuals. For example, we can read the r-labelled edge from (a, ε) to (b, ε) in both directions, i.e., $(a^{\mathcal{I}}, b^{\mathcal{I}}) = ((a, \varepsilon), (b, \varepsilon)) \in r^{\mathcal{I}}$ and $(b^{\mathcal{I}}, a^{\mathcal{I}}) = ((b, \varepsilon), (a, \varepsilon)) \in r^{-\mathcal{I}}$. The "shortcuts" due to transitive roles are shown as dashed lines, while the relationship between the nodes that represent ABox individuals is shown in grey. Please note that we did not indicate the interpretations of all concepts in the figure.

Since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, the elements of the domain are pairs (a, w), where a indicates the individual name that corresponds to the root of the tree, i.e., $a^{\mathcal{I}} = (a, \varepsilon)$ and the elements in the second place form a tree according to our definition of trees. For each individual name a in our ABox, we can, therefore, easily define the tree rooted in a as $\{w \mid (a, w) \in \Delta^{\mathcal{I}}\}$.

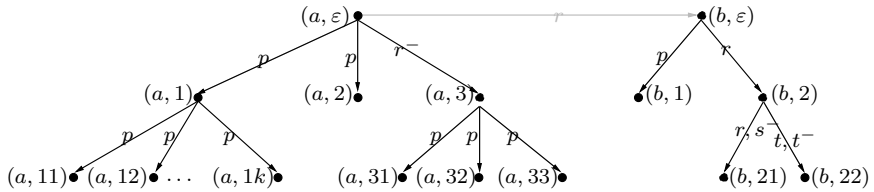

Figure 4.2: A forest base for the interpretation represented by Figure 4.1.

Figure 4.2 shows a representation of a forest base for the interpretation from Figure 4.1 above. For simplicity, the interpretation of concepts is no longer shown. The two trees, rooted in (a, ε) and (b, ε) respectively, are now clear.

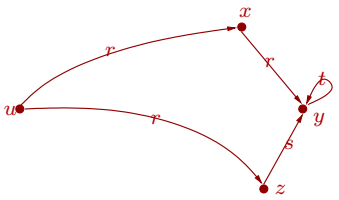

Figure 4.3: A graph representation of the query from Example 4.3.

A graphical representation of the query q from Example 4.3 is shown in Figure 4.3, where the meaning of the nodes and edges is analogous to the ones given for interpretations; e.g., the role conjunct $r(u, x)$ is represented by the r-labelled edge from the node u to the node x. We call this query a cyclic query since its underlying undirected graph is cyclic (cf. Definition 2.7).

Figure 4.4 shows a match π for q and $\mathcal{I}$ and we further have that $\mathcal{K} \models q$, i.e., the query is true in each model of the knowledge base.

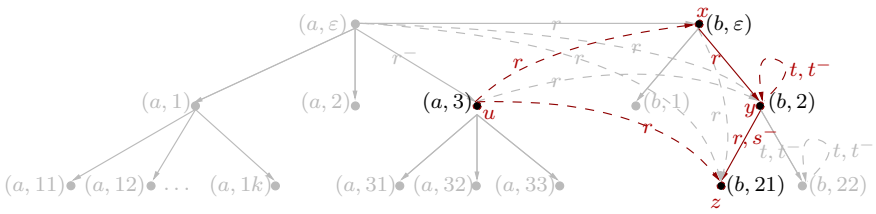

Figure 4.4: A match π for the query q from Example 4.3 onto the model $\mathcal{I}$ from Figure 4.1.

The forest model property is also exploited in the query rewriting process. We want to rewrite q into a set of queries $q_1, \ldots, q_n$ of ground or tree-shaped queries such that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_n$, i.e., each model of $\mathcal{K}$ satisfies at least on of the rewritten queries. Since the resulting queries are ground or tree-shaped, we can exploit known techniques for deciding entailment of these queries. As a first step, we transform q into a set of forest-shaped queries. Intuitively, forest-shaped queries consist of a set of tree-shaped sub-queries, where the roots of these trees might be arbitrarily interconnected (by conjuncts of the form $r(t, t')$). A tree-shaped query is a special case of a forest-shaped query where there is a single root node. We will call the arbitrarily interconnected terms of a forest-shaped query the root terms (or, for short, just roots). At the end of the rewriting process, we replace the root terms with individual names from $\mathsf{Inds}(\mathcal{A})$ and transform the tree parts into a concept by applying the rolling-up or tuple graph technique [22, 129].

In the proof of the correctness of our procedure, we use the structure of the forest bases in order to explicate the transitive "shortcuts" used in the query match. By explicating we mean that we replace each role conjunct that is mapped to such a shortcut with a sequence of role conjuncts such that an extended match for the modified query uses only paths that are in the forest base.

4.1.3 The Rewriting Steps

The rewriting process for a query q is a six stage process. At the end of this process, the rewritten query may or may not be in a forest shape. As we show later, this "don't know" non-determinism does not compromise the correctness of the algorithm. In the first stage, we derive a *collapsing* q_{co} of q by adding (possibly several) equality conjuncts to q. Consider, for example, the cyclic query $q = \{r(x, y), r(x, y'), s(y, z), s(y', z)\}$ (see Figure 4.5), which can be transformed into a tree-shaped one by adding the equality conjunct $y \approx y'$.

Figure 4.5: A representation of a cyclic query and of the tree-shaped query obtained by adding the conjunct $y \approx y'$ to the query depicted on the left hand side.

A common property of the next three rewriting steps is that they allow for substituting (implicit) shortcut edges with (explicit) paths that imply the shortcut. The three steps aim at different cases in which these shortcuts can occur and we describe their goals and application now in more detail.

The second stage is called *split rewriting*. In a split rewriting we take care of all role conjuncts that are matched to transitive shortcuts connecting elements of two different trees and by-passing one or both of their roots. We substitute these shortcuts with either one or two role conjuncts such that the roots are included. In our running example, π maps u to $(a, 3)$ and x to (b, ε). Hence $\mathcal{I} \models^{\pi} r(u, x)$, but the used r-edge is a transitive shortcut connecting the tree rooted in a with the tree rooted in b, and by-passing (a, ε). Similar arguments hold for the conjunct $r(u, z)$, where the path that implies this shortcut relationship goes via the two roots (a, ε) and (b, ε). It is clear that r must be a non-simple role since, in the forest base $\mathcal{J}$ for $\mathcal{I}$, there is no "direct" connection between different trees other than between the roots of the trees. Hence, $(\pi(u), \pi(x)) \in r^{\mathcal{I}}$ holds only because there is a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$. In the case of our example, r itself is transitive. A split rewriting eliminates transitive shortcuts between different trees of a canonical model and adds the "missing" variables and role conjuncts matching the sequence of edges that induce the shortcut. For the rewritten query, we also define a set of root terms, which contains the variables that are mapped to the roots in the canonical model. For our running example, we guess that the set of root terms is $\{ux, x\}$.

Figure 4.6 depicts the split rewriting

$$q_{sr} = \{\, r(u, ux), r(ux, x), r(x, y), t(y, y), s(z, y), r(x, z)\}$$

of q that is obtained from q by replacing (i) $r(u, x)$ with $r(u, ux)$ and $r(ux, x)$ and

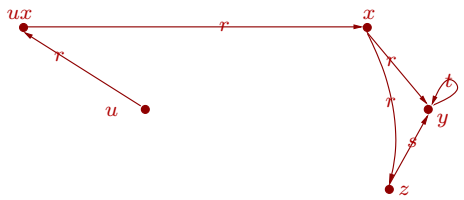

Figure 4.6: A split rewriting q_{sr} for the query shown in Figure 4.3.

(ii) $r(u, z)$ with $r(u, ux), r(ux, x)$, and $r(x, z)$. Please note that we both introduced a new variable (ux) and re-used an existing variable (x) and, since queries are sets, the conjuncts $r(u, ux)$ and $r(ux, x)$ occur only once in the rewritten query. Figure 4.7 shows a match for q_{sr} and the canonical model $\mathcal{I}$ of $\mathcal{K}$ in which the two trees are only connected via the roots.

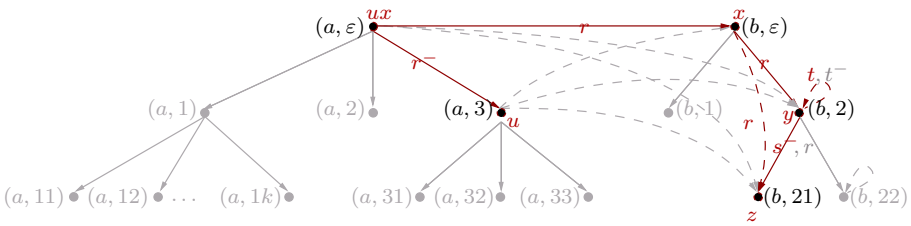

Figure 4.7: A split match π_{sr} for the query q_{sr} from Figure 4.7 onto the canonical interpretation from Figure 4.1.

Figure 4.7 also indicates that ux and x are the variables in q_{sr} that correspond to roots.

In the third step, called *loop rewriting*, we eliminate "loops" for variables v that do not correspond to roots by replacing conjuncts $r(v, v)$ with two conjunct $r(v, v')$ and $r(v', v)$, where v' can either be a new or an existing variable in q. In our running example, we eliminate the loop $t(y, y)$ as follows:

$$q_{\ell r} = \{\, r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), r(x, z)\}$$

is the query obtained from q_{sr} (see Figure 4.6) by replacing $t(y, y)$ with $t(y, y')$ and $t(y', y)$ for a new variable y'. Please note that, since t is defined as transitive and symmetric, $t(y, y)$ is still implied, i.e., the loop is also a transitive shortcut. Figure 4.8 shows the canonical interpretation $\mathcal{I}$ from Figure 4.1 with a match $\pi_{\ell r}$ for $q_{\ell r}$. The introduction of the new variable y' is needed in this case since there is no variable that could be re-used and the individual $(b, 22)$ is not in the range of the match π_{sr}.

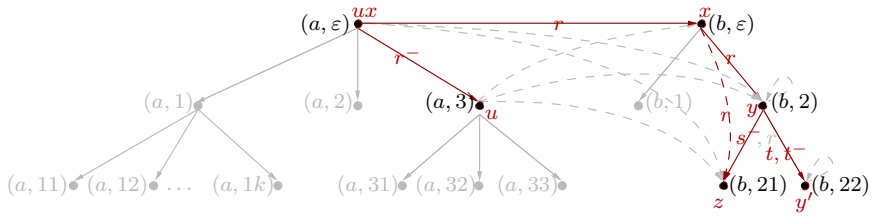

Figure 4.8: A loop rewriting $q_{\ell r}$ and a match for the canonical interpretation from Figure 4.1.

The forth rewriting step, called *forest rewriting*, again involves the replacement of role conjuncts with sets of role conjuncts. This allows for the elimination of cycles that are within a single tree. A forest rewriting q_{fr} for our example can be obtained from $q_{\ell r}$ by replacing the role conjunct $r(x, z)$ with $r(x, y)$ and $r(y, z)$, resulting in the query

$$q_{fr} = \{\, r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), r(x, y), r(y, z)\}.$$

Clearly, this results in tree-shaped sub-queries, one rooted in ux and one rooted in x. Hence q_{fr} is forest-shaped w.r.t. the root terms ux and x. Figure 4.9 shows the canonical interpretation $\mathcal{I}$ from Figure 4.1 with a match π_{fr} for q_{fr}.

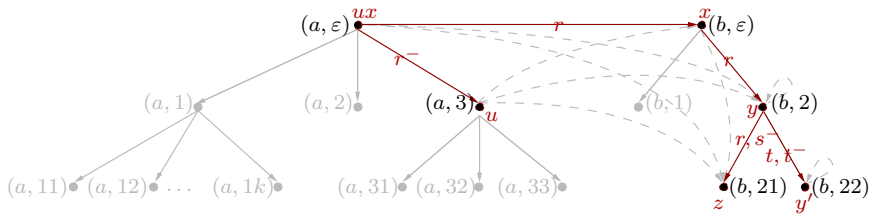

Figure 4.9: A forest rewriting q_{fr} and a forest match π_{fr} for the canonical interpretation from Figure 4.1.

In the fifth step, we use the standard rolling-up technique [22, 64] and express the tree-shaped sub-queries as concepts. In order to do this, we traverse each tree in a bottom-up fashion and replace each leaf (labelled with a concept C, say) and its incoming edge (labelled with a role r, say) with the concept $\exists r.C$ added to its predecessor.

During the reduction, we may introduce concepts that contain an intersection of roles. Intuitively, this occurs when the query contains several role conjuncts for the same pair of variables. We define, therefore, the extension of $\mathcal{SHIQ}$ with role conjunction/intersection, denoted as $\mathcal{SHIQ}^{\sqcap}$.

In addition to the constructors introduced for $\mathcal{SHIQ}$, $\mathcal{SHIQ}^{\sqcap}$ allows for concepts of the form

$$C ::= \forall R.C \mid \exists R.C \mid \leqslant n\ S.C \mid \geqslant n\ S.C,$$

where $R := r_1 \sqcap \ldots \sqcap r_n$, $S := s_1 \sqcap \ldots \sqcap s_n$, $r_1, \ldots, r_n$ are roles, and $s_1, \ldots, s_n$ are simple roles. The interpretation function is extended such that $(r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}} = {r_1}^{\mathcal{I}} \cap \ldots \cap {r_n}^{\mathcal{I}}$.

It is known that several expressive Description Logics can be extended with role conjunction [71, 72, 118, 142] and implementations of such logics are readily available. In the following section, we present a decision procedure for checking the consistency of $\mathcal{SHIQ}^{\sqcap}$ knowledge bases that also allows us to obtain worst-case optimal complexity bounds for query entailment in $\mathcal{SHIQ}$.

For example, the tree rooted in ux (i.e., the role conjunct $r(u, ux)$) can be replaced with the conjunct $(\exists r^{-}.\top)(ux)$. Similarly, the tree rooted in x (i.e., the role conjuncts $r(x, y), r(y, z), s(z, y), t(y, y')$, and $t(y', y)$) can be replaced with the conjunct

$$(\exists r.((\exists(r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists(t \sqcap \mathsf{Inv}(t)).\top))(x).$$

Please note that we have to use role conjunctions in the resulting query in order to capture the semantics of multiple role conjuncts relating the same pair of variables.

Recall that, in the split rewriting, we have guessed that x and ux correspond to roots and, therefore, correspond to individual names in $\mathsf{Inds}(\mathcal{A})$. In the sixth and last rewriting step, we guess which variable corresponds to which individual name and replace the variables with the guessed names. A possible guess for our running example would be that ux corresponds to a and x to b. This results in the (ground) query

$$\{(\exists r^{-}.\top)(a), r(a, b), (\exists r.((\exists(r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists(t \sqcap \mathsf{Inv}(t)).\top)))(b)\},$$

which is entailed by $\mathcal{K}$.

Please note that we focused in the running example on the most reasonable rewriting. There are several other possible rewritings, e.g., we obtain another rewriting from q_{fr} by replacing ux with b and x with a in the last step. For a UCQ, we apply the rewriting steps to each of the disjuncts separately.

At the end of the rewriting process, we have, for each disjunct, a set of ground queries and/or queries that were rolled-up into a single concept conjunct. The latter queries result from forest rewritings that are tree-shaped and have an empty set of roots. Such tree-shaped rewritings can match anywhere in a tree and can, thus, not be grounded. Finally, we check if our knowledge base entails the disjunction of all the rewritten queries. We show that there is a bound on the number of (forest-shaped) rewritings and hence on the number of queries produced in the rewriting process. Intuitively, such a bound exists since we give precise bounds on the number of role conjuncts and variables that can be introduced in the rewriting steps. Furthermore, the number of roles that can be used in the rewriting steps is bounded since our input knowledge base contains only a bounded number of roles.

Summing up, the rewriting process for a connected conjunctive query q involves the following steps:

1. Build all collapsings of q.
2. Build all split rewritings of each collapsing w.r.t. a subset R of roots.
3. Build all loop rewritings of the split rewritings.
4. Build all (forest-shaped) forest rewritings of the loop rewritings.
5. Roll-up each tree-shaped sub-query in a forest-rewriting into a concept conjunct and
6. replace the roots in R with individual names from the ABox in all possible ways.

Let $q_1, \ldots, q_n$ be the queries resulting from the rewriting process. In the next section, we define each rewriting step and prove that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \cdots \vee q_n$. Checking entailment for the rewritten queries can easily be reduced to KB consistency and any decision procedure for $\mathcal{SHIQ}^{\sqcap}$ KB consistency can be used in order to decide if $\mathcal{K} \models q$. We present one such decision procedure in Section 4.3.

4.2 Query Rewriting

In the previous section, we used several terms, e.g., tree- or forest-shaped query, rather informally. In the following, we give definitions for the terms used in the

query rewriting process. Once this is done, we formalise the query rewriting steps and prove the correctness of the procedure, i.e., we show that the forest-shaped queries obtained in the rewriting process can indeed be used for deciding whether a knowledge base entails the original query or not.

4.2.1 Tree- and Forest-Shaped Queries

In order to define tree- or forest-shaped queries more precisely, we use mappings between queries and trees or forests. In the following definition, we use the two relations $\bar{\in}$ and $\overset{*}{\approx}$, which were defined in Definition 2.5 on page 36.

Definition 4.4. Let q be a query. A *tree mapping* for q is a total function f from terms in q to a tree such that

1. for each $t, t' \in \mathsf{Terms}(q), t \overset{*}{\approx} t'$ iff $f(t) = f(t')$,
2. $r(t, t') \bar{\in} q$ iff $f(t)$ is a neighbour of $f(t')$, and,
3. if $a \in \mathsf{Inds}(q)$, $f(a) = \varepsilon$.

The query q is *tree-shaped* if $\sharp(\mathsf{Inds}(q)) \leq 1$ and there is a tree mapping for q.

A *root choice* R for q is a subset of $\mathsf{Terms}(q)$ such that $\mathsf{Inds}(q) \subseteq R$ and, if $t \in R$ and $t \overset{*}{\approx} t'$, then $t' \in R$. For $t \in R$, we use $\mathsf{Reach}(t)$ to denote the set of terms $t' \in \mathsf{Terms}(q)$ for which there exists a sequence of terms $t_1, \ldots, t_n \in \mathsf{Terms}(q)$ such that

1. $t_1 = t$ and $t_n = t'$,
2. for all $1 \leq i < n$, there is a role r such that $r(t_i, t_{i+1}) \bar{\in} q$, and,
3. for $1 < i \leq n$, if $t_i \in R$, then $t_i \overset{*}{\approx} t$.

We call R a *root splitting w.r.t. q* if either $R = \emptyset$ or if, for $t_i, t_j \in R$, $t_i \not\overset{*}{\approx} t_j$ implies that $\mathsf{Reach}(t_i) \cap \mathsf{Reach}(t_j) = \emptyset$. Each term $t \in R$ induces a sub-query

$$\begin{aligned} \mathsf{subq}(q, t) = \; & \{at \bar{\in} q \mid \text{ the terms in } at \text{ occur in } \mathsf{Reach}(t)\} \setminus \\ & \{r(t, t) \mid r(t, t) \bar{\in} q\}. \end{aligned}$$

A query q is *forest-shaped w.r.t. a root splitting R* if each sub-query $\mathsf{subq}(q, t)$ for $t \in R$ is tree-shaped or if $R = \emptyset$ and q is tree-shaped. △

For each term $t \in R$, we collect the terms that are reachable from t in the set $\mathsf{Reach}(t)$. By Condition 3, we make sure that R and $\overset{*}{\approx}$ are such that each $t' \in \mathsf{Reach}(t)$ is either not in R or $t \overset{*}{\approx} t'$. Since queries are connected by assumption, we would otherwise collect all terms in $\mathsf{Reach}(t)$ and not just those $t' \notin R$. For a root splitting, we require that the resulting sets are mutually disjoint for all terms $t, t' \in R$ that are not equivalent. This guarantees that all paths between the subqueries go via the root nodes of their respective trees. Intuitively, a forest-shaped query is one that can potentially be mapped onto a canonical interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that the terms in the root splitting R correspond to roots $(a, \varepsilon) \in \Delta^{\mathcal{I}}$. In the definition of $\mathsf{subq}(q, t)$, we exclude loops of the form $r(t, t) \bar{\in} q$, as these parts of the query are grounded later in the query rewriting process, and we allow arbitrary relationships between ground terms.

Consider, for example, the query q_{sr} of our running example from the previous section (cf. Figure 4.6, page 83). Let us again make the root choice $R = \{ux, x\}$ for q. The sets $\mathsf{Reach}(ux)$ and $\mathsf{Reach}(x)$ w.r.t. q_{sr} and R are $\{ux, u\}$ and $\{x, y, z\}$ respectively. Since the two sets are obviously disjoint, R is a root splitting w.r.t. q_{sr}. If we choose, however, $R = \{x, y\}$, the set R is not a root splitting w.r.t. q_{sr} since $\mathsf{Reach}(x) = \{ux, u, z\}$ and $\mathsf{Reach}(y) = \{z\}$ are not disjoint.

4.2.2 From Graphs to Forests

In this section, we give precise definition of the rewriting steps. Given an arbitrary query, we exhaustively apply the rewriting steps and show that we can use the resulting queries that are forest-shaped to decide entailment of the original query. Please note that the following definitions are for conjunctive queries and not for unions of conjunctive queries since we apply the rewriting steps for each disjunct separately.

Definition 4.5. Let q be a Boolean conjunctive query. A *collapsing* q_{co} of q is obtained by adding zero or more equality conjuncts of the form $t \approx t'$ for $t, t' \in \mathsf{Terms}(q)$ to q. We use $\mathsf{co}(q)$ to denote the set of all queries that are a collapsing of q.

Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base. A query q_{sr} is called a *split rewriting* of q w.r.t. $\mathcal{K}$ if it is obtained from q by choosing, for each conjunct $r(t, t') \bar{\in} q$, to either:

1. do nothing,

2. choose a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and replace $r(t, t')$ with $s(t, u)$, $s(u, t')$, or

3. choose a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and replace $r(t, t')$ with $s(t, u)$, $s(u, u')$, $s(u', t')$,

where $u, u' \in N_V$ are possibly fresh variables, i.e., variables that may or may not occur in q. We use $\mathsf{sr}_{\mathcal{K}}(q)$ to denote the set of all pairs (q_{sr}, R) for which there is a query $q_{co} \in \mathsf{co}(q)$ such that q_{sr} is a split rewriting of q_{co} and R is a root splitting w.r.t. q_{sr}.

A query $q_{\ell r}$ is called a *loop rewriting* of q w.r.t. a root splitting R and $\mathcal{K}$ if it is obtained from q by choosing, for all conjuncts of the form $r(t, t) \bar{\in} q$ with $t \notin R$, a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and by replacing $r(t, t)$ with two conjuncts $s(t, t')$ and $s(t', t)$ for $t' \in N_V$ a possibly fresh variable. We use $\mathsf{lr}_{\mathcal{K}}(q)$ to denote the set of all pairs $(q_{\ell r}, R)$ for which there is a tuple $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ such that $q_{\ell r}$ is a loop rewriting of q_{sr} w.r.t. R and $\mathcal{K}$.

For a forest rewriting, fix a set $V \subseteq N_V$ of variables not occurring in q such that $\sharp(V) \leq \sharp(\mathsf{Vars}(q))$. A *forest rewriting* q_{fr} w.r.t. a root splitting R of q and $\mathcal{K}$ is obtained from q by choosing, for each role conjunct $r(t, t')$ such that either $R = \emptyset$ and $r(t, t') \bar{\in} q$ or there is some $t_r \in R$ and $r(t, t') \bar{\in} \mathsf{subq}(q, t_r)$ to either

1. do nothing, or

2. choose a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and replace $r(t, t')$ with $\ell \leq \sharp(\mathsf{Vars}(q))$ role conjuncts $s(t_1, t_2), \ldots, s(t_\ell, t_{\ell+1})$, where $t_1 = t$, $t_{\ell+1} = t'$, and $t_2, \ldots, t_\ell \in \mathsf{Terms}(q) \cup V$.

We use $\mathsf{fr}_{\mathcal{K}}(q)$ to denote the set of all pairs (q_{fr}, R) for which there is a tuple $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ such that q_{fr} is a forest-shaped forest rewriting of $q_{\ell r}$ w.r.t. R and $\mathcal{K}$. △

If $\mathcal{K}$ is clear from the context, we say that q' is a split, loop, or forest rewriting of q instead of saying that q' is a split, loop, or forest rewriting of q w.r.t. $\mathcal{K}$. We assume that $\mathsf{sr}_{\mathcal{K}}(q), \mathsf{lr}_{\mathcal{K}}(q)$, and $\mathsf{fr}_{\mathcal{K}}(q)$ contain no isomorphic queries, i.e., differences in (newly introduced) variable names only are neglected. Please note that, if $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $r(t, t) \bar{\in} q_{sr}$ with $t \notin R$, this does not necessarily imply that a loop rewriting can be applied to the conjunct $r(t, t)$ since r could be a simple role.

In Section 4.2.5, we give detailed proofs of the correctness of the rewriting steps. The proofs also show why the bounds on the number of new variables and the number of newly introduced role atoms suffice.

In the next section, we show how we can build a disjunction of conjunctive queries $q_1 \vee \cdots \vee q_\ell$ from the queries in $\mathsf{fr}_\mathcal{K}(q)$ such that each q_i for $1 \leq i \leq \ell$ is either of the form $C(v)$ for a single variable $v \in \mathsf{Vars}(q_i)$ or q_i is ground, i.e., q_i contains only constants and no variables. It then remains to show that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \cdots \vee q_\ell$.

4.2.3 From Trees to Concepts

In order to transform a tree-shaped query into a single concept conjunct and a forest-shaped query into a ground query, we define a mapping f from the terms in each tree-shaped sub-query to a tree. We then incrementally build a concept that corresponds to the tree-shaped query by traversing the tree in a bottom-up fashion, i.e., from the leaves upwards to the root. This technique for expressing tree-shaped queries as concepts is well-known and is usually called the rolling-up or tuple graph technique [22, 129].

Definition 4.6. Let q be a tree-shaped query with at most one individual name. If $a \in \mathsf{Inds}(q)$, then let $t_r = a$ otherwise let $t_r = v$ for some variable $v \in \mathsf{Vars}(q)$. We now inductively assign, to each term $t \in \mathsf{Terms}(q)$, a concept $\mathsf{con}(q, t)$ as follows:

- Let f be a tree mapping such that $f(t_r) = \varepsilon$.
- If $f(t)$ is a leaf, then $\mathsf{con}(q, t) = \bigsqcap_{C(t) \bar{\in} q} C$.
- If $f(t)$ has successors $f(t_1), \ldots, f(t_k)$, then

$$\mathsf{con}(q, t, f) = \bigsqcap_{C(t) \bar{\in} q} C \sqcap \bigsqcap_{1 \leq i \leq k} \exists \big(\bigsqcap_{r(t,t_i) \bar{\in} q} r \big).\mathsf{con}(q, t_i).$$

Finally, the *query concept of q w.r.t. t_r* is $\mathsf{con}(q, t_r)$. △

Please note that the above definition takes equality conjuncts into account. This is because the function f is bijective modulo $\overset{*}{\approx}$ and, in case there are concept conjuncts $C(t)$ and $C(t')$ for $t \overset{*}{\approx} t'$, both concepts are conjoined in the query

concept due to the use of the relation $\bar{\in}$. Similar arguments can be applied to the role conjuncts.

The following lemma shows that query concepts indeed capture the semantics of q.

Lemma 4.7. *Let q be a tree-shaped query with $t_r \in \mathsf{Terms}(q)$ as defined above, $C_q = \mathsf{con}(q, t_r)$, and $\mathcal{I}$ an interpretation. Then $\mathcal{I} \models q$ iff there is a match π and an element $d \in {C_q}^{\mathcal{I}}$ such that $\pi(t_r) = d$.*

The proof given by [65] easily transfers from $\mathcal{DLR}$ to $\mathcal{SHIQ}$. By applying the result from the above lemma, we can now transform a forest-shaped query into a ground query as follows:

Definition 4.8. Let $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ for $R \neq \emptyset$, and $\tau\colon R \to \mathsf{Inds}(\mathcal{A})$ a total function such that, for each $a \in \mathsf{Inds}(q)$, $\tau(a) = a$ and, for $t, t' \in R$, $\tau(t) = \tau(t')$ iff $t \overset{*}{\approx} t'$. We call such a mapping τ a *ground mapping for R w.r.t. $\mathcal{A}$*. We obtain a ground query $\mathsf{ground}(q_{fr}, R, \tau)$ of q_{fr} w.r.t. the root splitting R and ground mapping τ as follows:

- replace each $t \in R$ with $\tau(t)$, and,
- for each $a \in \mathsf{ran}(\tau)$, replace the sub-query $q_a = \mathsf{subq}(q_{fr}, a)$ with $\mathsf{con}(q_a, a)$.

We define the set $\mathsf{ground}_{\mathcal{K}}(q)$ of *ground queries for q* w.r.t. $\mathcal{K}$ as follows:

$$\mathsf{ground}_{\mathcal{K}}(q) = \{q' \mid \begin{array}{l}\text{there exists some } (q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q) \text{ with } R \neq \emptyset \\ \text{and some ground mapping } \tau \text{ w.r.t. } \mathcal{A} \text{ and } R \\ \text{such that } q' = \mathsf{ground}(q_{fr}, R, \tau)\}\end{array}$$

We define the set $\mathsf{tree}_{\mathcal{K}}(q)$ of *tree queries for q* as follows:

$$\mathsf{tree}_{\mathcal{K}}(q) = \{q' \mid \begin{array}{l}\text{there exists some } (q_{fr}, \emptyset) \in \mathsf{fr}_{\mathcal{K}}(q) \text{ and} \\ v \in \mathsf{Vars}(q_{fr}) \text{ such that } q' = (\mathsf{con}(q_{fr}, v))(v)\}\end{array}$$

△

Going back to our running example, we have already seen that $(q_{fr}, \{ux, x\}) \in \mathsf{fr}_{\mathcal{K}}(q)$ for

$$q_{fr} = \{r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), r(y, z)\}.$$

There are also several other queries in the set $\mathsf{fr}_{\mathcal{K}}(q)$, e.g., $(q, \{u, x, y, z\})$, where q is the original query and the root splitting R is such that $R = \mathsf{Terms}(q)$, i.e., all terms are in the root choice for q. In order to build the set $\mathsf{ground}_{\mathcal{K}}(q)$, we now build all possible ground mappings τ for the set $\mathsf{Inds}(\mathcal{A})$ of individual names in our ABox and the root splittings for the queries in $\mathsf{fr}_{\mathcal{K}}(q)$. The tuple $(q_{fr}, \{ux, x\}) \in \mathsf{fr}_{\mathcal{K}}(q)$ contributes two ground queries for the set $\mathsf{ground}_{\mathcal{K}}(q)$:

$$\mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto a, x \mapsto b\}) =$$
$$\{r(a, b), (\exists \mathsf{Inv}(r).\top)(a), (\exists r.(\exists(r \sqcap \mathsf{Inv}(s)).\top \sqcap \exists(t \sqcap \mathsf{Inv}(t)).\top))(b)\},$$

where $\exists \mathsf{Inv}(r).\top$ is the query concept for the (tree-shaped) sub-query $\mathsf{subq}(q_{fr}, ux)$ and $\exists r.(\exists(r \sqcap \mathsf{Inv}(s)).\top \sqcap \exists(t \sqcap \mathsf{Inv}(t)).\top)$ is the query concept for $\mathsf{subq}(q_{fr}, x)$ and

$$\mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto b, x \mapsto a\}) =$$
$$\{r(b, a), (\exists \mathsf{Inv}(r).\top)(b), (\exists r.(\exists(r \sqcap \mathsf{Inv}(s)).\top \sqcap \exists(t \sqcap \mathsf{Inv}(t)).\top))(a)\}.$$

The tuple $(q, \{u, x, y, z\}) \in \mathsf{fr}_{\mathcal{K}}(q)$, however, does not contribute a ground query since, for a ground mapping, we require that $\tau(t) = \tau(t')$ iff $t \overset{*}{\approx} t'$ and there are only two individual names in $\mathsf{Inds}(\mathcal{A})$ compared to four terms q that need a distinct value. Intuitively, this does not result in any loss of generality since in the first rewriting step (collapsing) we produce all those queries in which the terms of q have been identified with each other in all possible ways. In our example, $\mathcal{K} \models q$ and $\mathcal{K} \models q_1 \vee \cdots \vee q_\ell$, where $q_1 \vee \cdots \vee q_\ell$ are the queries from $\mathsf{tree}_{\mathcal{K}}(q)$ and $\mathsf{ground}_{\mathcal{K}}(q)$ since each model $\mathcal{I}$ of $\mathcal{K}$ satisfies $q_i = \mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto a, x \mapsto b\})$.

4.2.4 Query Matches

In this section, we define certain classes of matches for a query into a model. We use these special matches in the next section, when we prove the correctness of our procedure.

Even if a query is true in a canonical model, it does not necessarily mean that the query is tree- or forest-shaped. However, a match π for a canonical interpretation can guide the process of rewriting a query. Similarly to the definition of tree- or forest-shaped queries, we define the shape of matches for a query. In particular, we introduce three different kinds of matches: *split matches, forest matches*, and *tree matches* such that every tree match is a forest match, and every forest match is a split match. The correspondence to the query shapes is as

follows: given a split match π, the set of all root nodes (a, ε) in the range of the match define a root splitting for the query; if π is additionally a forest match, the query is forest-shaped w.r.t. the root splitting induced by π and, if π is additionally a tree match, then the whole query can be mapped to a single tree (i.e., the query is tree-shaped or forest-shaped w.r.t. an empty root splitting). Given an arbitrary query match into a canonical model, we can first obtain a split match and then a tree or forest match by using the structure of the canonical model to guide the application of the rewriting steps.

Definition 4.9. Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base, q a query, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a canonical model of $\mathcal{K}$, and $\pi \colon \mathsf{Terms}(q) \to \Delta^{\mathcal{I}}$ an evaluation such that $\mathcal{I} \models^{\pi} q$. We call π a *split match* if, for all $r(t, t') \bar{\in} q$, one of the following holds:

1. $\pi(t) = (a, \varepsilon)$ and $\pi(t') = (b, \varepsilon)$ for some $a, b \in \mathsf{Inds}(\mathcal{A})$; or
2. $\pi(t) = (a, w)$ and $\pi(t') = (a, w')$ for some $a \in \mathsf{Inds}(\mathcal{A})$ and $w, w' \in \mathbb{N}^*$.

We call π a *forest match* if, additionally, for each term $t_r \in \mathsf{Terms}(q)$ with $\pi(t_r) = (a, \varepsilon)$ and $a \in \mathsf{Inds}(\mathcal{A})$, there is a total and bijective mapping f from $\{(a, w) \mid (a, w) \in \mathsf{ran}(\pi)\}$ to a tree T such that $r(t, t') \bar{\in} \mathsf{subq}(q, t_r)$ implies that $f(\pi(t))$ is a neighbour of $f(\pi(t'))$. We call π a *tree match* if, additionally, there is an $a \in \mathsf{Inds}(\mathcal{A})$ such that each element in $\mathsf{ran}(\pi)$ is of the form (a, w).

A split match π for a canonical interpretation induces a (possibly empty) root splitting R such that $t \in R$ iff $\pi(t) = (a, \varepsilon)$ for some $a \in \mathsf{Inds}(\mathcal{A})$. We call R the *root splitting induced by* π. △

For two elements (a, w) and (a, w') in a canonical model, the path from (a, w) to (a, w') is the sequence $(a, w_1), \ldots, (a, w_n)$ where $w = w_1, w' = w_n$, and, for $1 \leq i < n, w_i$ is a predecessor of w_{i+1}. The length of the path is n. Please note that, for a forest match, we do not require that w is a neighbour of w' or vice versa. This still allows to map role conjuncts to paths in the canonical model of length greater than two, but such paths must be between ancestors and not between elements in different branches of the tree. The mapping f to a tree also makes sure that, if R is the induced root splitting, then each sub-query $\mathsf{subq}(q, t)$ for $t \in R$ is tree-shaped. For a tree match, the root splitting is either empty or $t \overset{*}{\approx} t'$ for each $t, t' \in R$, i.e., there is a single root modulo $\overset{*}{\approx}$, and the whole query is tree-shaped.

4.2.5 Correctness of the Query Rewriting

The following lemmas state the correctness of the rewriting step by step for each of the rewriting stages. As motivated in the previous section, we can use a given canonical model to guide the rewriting process such that we obtain a forest-shaped query that also has a match into the model.

Lemma 4.10. *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I} \models q$, then there is a collapsing q_{co} of q such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for π_{co} an injection modulo $\overset{*}{\approx}$.*

2. *If $\mathcal{I} \models^{\pi_{co}} q_{co}$ for a collapsing q_{co} of q, then $\mathcal{I} \models q$.*

Proof. For (1), let π be such that $\mathcal{I} \models^{\pi} q$, let q_{co} be the collapsing of q that is obtained by adding an conjunct $t \approx t'$ for all terms $t, t' \in \mathsf{Terms}(q)$ for which $\pi(t) = \pi(t')$. By definition of the semantics, $\mathcal{I} \models^{\pi} q_{co}$ and π is an injection modulo $\overset{*}{\approx}$.

Condition (2) trivially holds since $q \subseteq q_{co}$ and hence $\mathcal{I} \models^{\pi_{co}} q$. □

In the next step, we show that there is a split rewriting for at least one of the collapsings.

Lemma 4.11. *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models^{\pi} q$, then there is a pair $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and a split match π_{sr} such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, R is the induced root splitting of π_{sr}, and π_{sr} is an injection modulo $\overset{*}{\approx}$.*

2. *If $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{sr}} q_{sr}$ for some match π_{sr}, then $\mathcal{I} \models q$.*

For the first part of the lemma, we proceed exactly as illustrated in the example section and use the canonical model $\mathcal{I}$ and the match π to guide the rewriting steps. We first build a collapsing $q_{co} \in \mathsf{co}(q)$ as described in the proof of Lemma 4.10 such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for π_{co} an injection modulo $\overset{*}{\approx}$. Since $\mathcal{I}$ is canonical, paths between different trees can only occur due to non-simple roles, and thus we can replace each role conjunct that uses such a shortcut with two or three role conjuncts such that these roots are explicitly included in the query (cf. the query and match in Figure 4.4 on page 81 and the obtained split rewriting and split match in Figure 4.7 on page 83).

Proof. The proof of 2 is relatively straightforward: since $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$, there is a collapsing q_{co} of q such that q_{sr} is a split rewriting of q_{co}. All roles replaced in a split rewriting are non-simple and are replaced with transitive subroles. Since $\mathcal{I} \models q_{sr}$ by assumption, we have, therefore, that $\mathcal{I} \models q_{co}$. By Lemma 4.10 (2), we then have that $\mathcal{I} \models q$ as required.

We go through the proof of (1) in more detail: let q_{co} be in $\mathsf{co}(q)$ such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for a match π_{co} that is injective modulo $\overset{*}{\approx}$. Such a collapsing q_{co} and match π_{co} exist due to Lemma 4.10. If π_{co} is a split match w.r.t. q and $\mathcal{I}$ already, we are done since a split match induces a root splitting R and (q_{co}, R) is trivially in $\mathsf{sr}_{\mathcal{K}}(q)$. If π_{co} is not a split match, there are at least two terms t, t' with $r(t, t') \bar{\in} q_{co}$ such that $\pi_{co}(t) = (a, w), \pi_{co}(t') = (a', w'), a \neq a'$, and $w \neq \varepsilon$ or $w' \neq \varepsilon$. We distinguish two cases:

1. Both t and t' are not mapped to roots, i.e., $w \neq \varepsilon$ and $w' \neq \varepsilon$. Since $\mathcal{I} \models^{\pi_{co}} r(t, t')$, we have that $(\pi_{co}(t), \pi_{co}(t')) \in r^{\mathcal{I}}$. Since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, there must be a role s with $s \sqsubseteq^{*}_{\mathcal{R}} r$ and $s \in \mathsf{Trans}_{\mathcal{R}}$ such that

$$\{(\pi_{co}(t), (a, \varepsilon)),\ ((a, \varepsilon), (a', \varepsilon)),\ ((a', \varepsilon), \pi_{co}(t'))\} \subseteq s^{\mathcal{I}}.$$

   If there is some $\hat{t} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t}) = (a, \varepsilon)$, then let $u = \hat{t}$, otherwise let u be a fresh variable. Similarly, if there is some $\hat{t'} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t'}) = (a', \varepsilon)$, then let $u' = \hat{t'}$, otherwise let u' be a fresh variable. Hence, we can define a split rewriting q_{sr} of q_{co} by replacing $r(t, t')$ with $s(t, u)$, $s(u, u')$, and $s(u', t')$. We then define a new mapping π_{sr} that agrees with π_{co} on all terms that occur in q_{co} and that maps u to (a, ε) and u' to (a', ε).

2. Either t or t' is mapped to a root. Without loss of generality, let this be t, i.e., $\pi(t) = (a, \varepsilon)$. We can use the same arguments as above: since $\mathcal{I} \models^{\pi_{co}} r(t, t')$, we have that $(\pi(t), \pi(t')) \in r^{\mathcal{I}}$ and, since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, there must be a role s with $s \sqsubseteq^{*}_{\mathcal{R}} r$ and $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $\{(\pi(t), (a', \varepsilon)), ((a', \varepsilon), \pi(t'))\} \subseteq s^{\mathcal{I}}$. If there is some $\hat{t} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t}) = (a', \varepsilon)$, then let $u = \hat{t}$, otherwise let u be a fresh variable. We then define a split rewriting q_{sr} of q_{co} by replacing $r(t, t')$ with $s(t, u), s(u, t')$ and a mapping π_{sr} that agrees with π_{co} on all terms that occur in q_{co} and that maps u to (a', ε).

It immediately follows that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$. We can proceed as described above for

each role conjunct $r(t, t')$ for which $\pi(t) = (a, w)$ and $\pi(t') = (a', w')$ with $a \neq a'$ and $w \neq \varepsilon$ or $w' \neq \varepsilon$. This will result in a split rewriting q_{sr} and a split match π_{sr} such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$. Furthermore, π_{sr} is injective modulo $\overset{*}{\approx}$ since we only introduce new variables when the variable is mapped to an element that is not yet in the range of the match. Since π_{sr} is a split match, it induces a root splitting R and, hence, $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ as required. □

Lemma 4.12. *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and a mapping $\pi_{\ell r}$ such that $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $\pi_{\ell r}$ is an injection modulo $\overset{*}{\approx}$, R is the root splitting induced by $\pi_{\ell r}$ and, for each $r(t, t) \bar{\in} q_{\ell r}$, $t \in R$.*

2. *If $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$ for some match $\pi_{\ell r}$, then $\mathcal{I} \models q$.*

For the first part, we proceed again as described in the examples section and use the canonical model $\mathcal{I}$ and the match π to guide the rewriting process. We first build a split rewriting q_{sr} and its root splitting R as described in the proof of Lemma 4.11 such that $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{sr}} q_{sr}$ for a split match π_{sr}. Since $\mathcal{I}$ is a canonical model, it has a forest base $\mathcal{J}$. In a forest base, non-root nodes cannot be successors of themselves, so each such loop is a shortcut due to some transitive role. An element that is say r-related to itself has, therefore, a neighbour that is both an r- and $\mathsf{Inv}(r)$-successor. Depending on whether this neighbour is already in the range of the match, we can re-use an existing variable or introduce a new one, when making this path explicit (cf. the loop rewriting depicted in Figure 4.8 on page 84 obtained from the split rewriting shown in Figure 4.7 on page 83).

Proof. The proof of (2) is analogous to the one given in Lemma 4.11 since, by definition of loop rewritings, all roles replaced in a loop rewriting are again non-simple.

For (1), let $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ be such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, π_{sr} is a split match, and R is the root splitting induced by π_{sr}. Such a split rewriting q_{sr} and match π_{sr} exist due to Lemma 4.11 and the canonicity of $\mathcal{I}$.

Let $r(t, t) \bar{\in} q_{sr}$ for $t \notin R$. Since R is the root splitting induced by π_{sr} and since $t \notin R$, $\pi_{sr}(t) = (a, w)$ for some $a \in \mathsf{Inds}(\mathcal{A})$ and $w \neq \varepsilon$. Now, let $\mathcal{J}$ be a forest base for $\mathcal{I}$. We show that there exists a neighbour d of $\pi_{sr}(t)$ and a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and $(\pi_{sr}(t), d) \in s^{\mathcal{I}} \cap \mathsf{Inv}(s)^{\mathcal{I}}$. Since $\mathcal{I} \models^{\pi_{sr}} q_{sr}$,

we have $(\pi_{sr}(t), \pi_{sr}(t)) \in r^{\mathcal{I}}$. Since $\mathcal{J}$ is a forest base and since $w \neq \varepsilon$, we have $(\pi_{sr}(t), \pi_{sr}(t)) \notin r^{\mathcal{J}}$. It follows that there is a sequence $d_1, \dots, d_n \in \Delta^{\mathcal{I}}$ and a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$, $d_1 = \pi_{sr}(t) = d_n$, and $(d_i, d_{i+1}) \in s^{\mathcal{J}}$ for $1 \leq i < n$ and $d_i \neq d_1$ for each i with $1 < i < n$. Then it is not hard to see that, because $\{w' \mid (a, w') \in \Delta^{\mathcal{I}}\}$ is a tree and $w \neq \varepsilon$, we can choose d_i such that $d_2 = d_{n-1}$. Since $(d_1, d_2) \in s^{\mathcal{J}}$ and $(d_{n-1}, d_n) \in s^{\mathcal{J}}$ with $d_{n-1} = d_2$ and $d_n = d_1$, the role s and the element $d = d_2$ is as required. For each $r(t,t) \bar{\in} q_{sr}$ with $t \notin R$, select an element $d_{r,t}$ and a role $s_{r,t}$ as described above. Now let $q_{\ell r}$ be obtained from q_{sr} by doing the following for each $r(t,t) \bar{\in} q_{sr}$ with $t \notin R$:

- if $d_{r,t} = \pi_{sr}(t')$ for some $t' \in \mathsf{Terms}(q_{sr})$, then replace $r(t,t)$ with $s_{r,t}(t,t')$ and $s_{r,t}(t',t)$;
- otherwise, introduce a new variable $v_{r,t} \in N_V$ and replace $r(t,t)$ with $s_{r,t}(t, v_{r,t})$ and $s_{r,t}(v_{r,t}, t)$.

Let $\pi_{\ell r}$ be obtained from π_{sr} by extending it with $\pi_{\ell r}(v_{r,t}) = d_{r,t}$ for each newly introduced variable $v_{r,t}$. By definition of $q_{\ell r}$ and $\pi_{\ell r}$, $q_{\ell r}$ is connected, $\pi_{\ell r}$ is injective modulo $\overset{*}{\approx}$, and $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$. □

The following lemma shows that we indeed restrict our attention to forest matches and forest-shaped queries:

Lemma 4.13. *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for a forest match π_{fr}, R is the induced root splitting of π_{fr}, and π_{fr} is an injection modulo $\overset{*}{\approx}$.*
2. *If $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for some match π_{fr}, then $\mathcal{I} \models q$.*

The main challenge is again the proof of (1) and, before we prove the lemma, we give a short and informal idea of the proof. At this point, we know from Lemma 4.12 that we can use a query $q_{\ell r}$ for which there is a root splitting R and a split match $\pi_{\ell r}$. Since $\pi_{\ell r}$ is a split match, the match for each such sub-query is restricted to a tree and thus we can separately transform each sub-query of $q_{\ell r}$ induced by a term in the root choice. The following example is meant to illustrate why the given bounds on the numbers of new variables and role conjuncts that can be introduced in a forest rewriting suffice. Figure 4.10 depicts the representation

of a tree from a canonical model, where we use only the second part of the names for the elements, e.g., we use just ε instead of (a, ε). For simplicity, we also do not indicate the concepts and roles that label the nodes and edges, respectively. We use black color to indicate the nodes and edges that are used in the match for a query and dashed lines for shortcuts due to transitive roles. In the example, the grey edges are those that belong to the forest base and the query match uses only shortcuts.

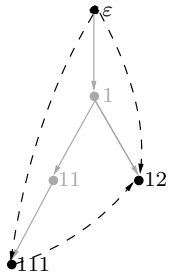

Figure 4.10: A part of a representation of a canonical model, where the black nodes and edges are used in a match for a query and dashed edges indicate shortcuts due to transitive roles.

The forest rewriting aims at making the shortcuts more explicit by replacing them with as few edges as necessary to obtain a tree match. In order to do this, we need to include the "common ancestors" in the forest base between each two nodes used in the match. For $w, w' \in \mathbb{N}^*$, we therefore define the *longest common prefix* (LCP) of w and w' as the longest $\hat{w} \in \mathbb{N}^*$ such that $\hat{w}$ is a prefix of both w and w'. For a forest rewriting, we now determine the LCPs of any two nodes in the range of the match and add a variable for those LCPs that are not yet in the range of the match to the set V of new variables used in the forest rewriting. In the example from Figure 4.10 the set V contains a single variable v_1 for the node 1.

We now explicate the shortcuts as follows: for any edge used in the match, e.g., the edge from ε to 111 in the example, we define its *path* as the sequence of elements on the path in the forest base, e.g., the path for the edge from ε to 111 is $\varepsilon, 1, 11, 111$. The *relevant path* is obtained by dropping all elements from the path that are not in the range of the mapping or correspond to a variable in the set V, resulting in a relevant path of $\varepsilon, 1, 111$ for the example. We now replace the role conjunct that was matched to the edge from ε to 111 with two role conjuncts such that the match uses the edge from ε to 1 and from 1 to 111. An appropriate transitive sub-role exists since otherwise there could not be a shortcut. Similar

arguments can be used to replace the role conjunct mapped to the edge from 111 to 12 and for the one that is mapped to the edge from ε to 12, resulting in a match as represented by Figure 4.11. The given restriction on the cardinality of the set V is no limitation since the number of LCPs in the set V is maximal if there is no pair of nodes such that one is an ancestor of the other. Since a tree of n leaf nodes that is at least binarily branching can have at most n inner nodes, we need at most n new variables for a query in n variables.

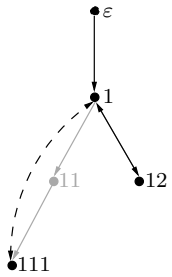

Figure 4.11: The match for a forest rewriting obtained from the example given in Figure 4.10.

For the bound on the number of role conjuncts that can be used in the replacement of a single role conjunct, consider, for example, the cyclic query

$$q = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_4)\},$$

for the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ with $\mathcal{T} = \emptyset, \mathcal{R} = \{r \sqsubseteq t\}$ with $t \in \mathsf{Trans}_{\mathcal{R}}$ and $\mathcal{A} = \{(\exists r.(\exists r.(\exists r.\top)))(a)\}$. It is not hard to check that $\mathcal{K} \models q$. Similarly to our running example from the previous section, there is also a single rewriting that is true in each canonical model of the KB, which is obtained by building only a forest rewriting and doing nothing in the other rewriting steps, except for choosing the empty set as root splitting in the split rewriting step. In the forest rewriting, we can explicate the shortcut used in the mapping for $t(x_1, x_4)$ by replacing $t(x_1, x_4)$ with $t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)$.

Now that we have informally introduced the main ideas, we prove the above lemma:

Proof. The proof of (2) is again analogous to the one given in Lemma 4.11. For (1), let $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ be such that $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, R is the root splitting induced by $\pi_{\ell r}$, $\pi_{\ell r}$ is injective modulo $\overset{*}{\approx}$ and, for each $r(t, t) \bar{\in} q_{\ell r}$, $t \in R$. Such a loop

rewriting and match $\pi_{\ell r}$ exist due to Lemma 4.12 and the canonicity of $\mathcal{I}$. By definition, R is a root splitting w.r.t. $q_{\ell r}$ and $\mathcal{K}$.

For $w, w' \in \mathbb{N}^*$, the *longest common prefix* (LCP) of w, w' is the longest $\hat{w} \in \mathbb{N}^*$ such that $\hat{w}$ is prefix of both w and w'. For the match $\pi_{\ell r}$ we now define the set D as follows:

$$D = \mathsf{ran}(\pi_{\ell r}) \cup \{(a, w) \in \Delta^{\mathcal{I}} \mid \begin{array}{l} w \text{ is the LCP of some } w, w' \\ \text{with } (a, w'), (a, w'') \in \mathsf{ran}(\pi_{\ell r})\}. \end{array}$$

Let $V \subseteq N_V \setminus \mathsf{Vars}(q_{\ell r})$ be such that, for each $d \in D \setminus \mathsf{ran}(\pi_{\ell r})$, there is a unique $v_d \in V$. We now define a mapping π_{fr} as $\pi_{\ell r} \cup \{v_d \mapsto d \mid v_d \in V \text{ and } d \in D \setminus \mathsf{ran}(\pi_{\ell r})\}$. By definition of V and v_d, π_{fr} is a loop match as well. The set $V \cup \mathsf{Vars}(q_{\ell r})$ will be the set of variables for the new query q_{fr}. Note that $\mathsf{ran}(\pi_{fr}) = D$.

Fact (a) if $(a, w), (a, w') \in \mathsf{ran}(\pi_{fr})$, then $(a, w'') \in \mathsf{ran}(\pi_{fr})$, where w'' is the LCP of w and w';

Fact (b) $\sharp(V) \leq \sharp(\mathsf{Vars}(q_{\ell r}))$

Fact (b) is due to the fact that, in the worst case, all (a, w) in $\mathsf{ran}(\pi_{\ell r})$ are "incomparable" and can thus be seen as leaves of a binarily branching tree. Now, a tree that has n leaves and is at least binarily branching at every non-leaf has at most n inner nodes, and thus $\sharp(V) \leq \sharp(\mathsf{Vars}(q_{\ell r}))$.

For a pair of individuals $d, d' \in \Delta^{\mathcal{I}}$, the *path* from d to d' is the (unique) shortest sequence of elements $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$ such that $d_1 = d$, $d_n = d'$, and d_{i+1} is a neighbour of d_i for all $1 \leq i < n$. The *length of a path* is the number of elements in it, i.e., the path $d_1, \ldots, d_n$ is of length n. The *relevant path* $d'_1, \ldots, d'_\ell$ from d to d' is the sub-sequence of $d_1, \ldots, d_n$ that is obtained by dropping all elements $d_i \notin D$.

Claim 1. Let $r(t, t') \bar{\in} \mathsf{subq}(q_{\ell r}, t_r)$ for some $t_r \in R$ and let $d'_1, \ldots, d'_\ell$ be the relevant path from $d = d'_1 = \pi_{\ell r}(t)$ to $d' = d'_\ell = \pi_{\ell r}(t')$. If $\ell > 2$, there is a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^*_{\mathcal{R}} r$ and $(d'_i, d'_{i+1}) \in s^{\mathcal{I}}$ for all $1 \leq i < \ell$.

Proof. Let $d_1, \ldots, d_n$ be the path and $d'_1, \ldots, d'_\ell$ the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$. Then $\ell > 2$ implies $n > 2$. We have to show that there is a role s as in the claim. Let $\mathcal{J}$ be a forest base for $\mathcal{I}$. Since $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $n > 2$ implies $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^{\mathcal{I}} \setminus r^{\mathcal{J}}$. Since $\mathcal{I}$ is based on $\mathcal{J}$, it follows that there

is an $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$, and $(d_i, d_{i+1}) \in s^{\mathcal{J}}$ for all $1 \leq i < n$. By construction of $\mathcal{I}$ from $\mathcal{J}$, it follows that $(d'_i, d'_{i+1}) \in s^{\mathcal{I}}$ for all $1 \leq i < \ell$, which finishes the proof of the claim.

Now let q_{fr} be obtained from $q_{\ell r}$ as follows: for each role conjunct $r(t, t) \bar{\in} \mathsf{subq}(q_{\ell r}, t_r)$ with $t_r \in R$, if the length of the relevant path $d'_1, \ldots, d'_\ell$ from $d'_1 = \pi_{\ell r}(t)$ to $d'_\ell = \pi_{\ell r}(t')$ is greater than 2, then select a role s and variables $t_j \in D$ such that $\pi_{fr}(t_j) = d'_j$ as in Claim 1 and replace the conjunct $r(t, t')$ with $s(t_1, t_2), \ldots, s(t_{\ell-1}, t_\ell)$, where $t = t_1$, $t' = t_\ell$. Please note that these t_j can be chosen in a "don't care" non-deterministic way since π_{fr} is injective modulo $\overset{*}{\approx}$, i.e., if $\pi_{fr}(t_j) = d_j = \pi_{fr}(t'_j)$, then $t_j \overset{*}{\approx} t'_j$ and we can pick any of these.

We now have to show that

(i) $\mathcal{I} \models^{\pi_{fr}} q_{fr}$, and

(ii) π_{fr} is a forest match.

For (i), let $r(t, t') \bar{\in} q_{\ell r} \setminus q_{fr}$ and let $s(t_1, t_2), \ldots, s(t_{\ell-1}, t_\ell)$ be the conjuncts in q_{fr} that replace $r(t, t')$ in q_{lr}. Since $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $\mathcal{I} \models^{\pi_{\ell r}} r(t, t')$ and $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^{\mathcal{I}}$. Since $r(t, t')$ was replaced in q_{fr}, the length of the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$ is greater than 2. Hence, it must be the case that $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^{\mathcal{I}} \setminus r^{\mathcal{J}}$. Let $d_1, \ldots, d_n$ with $d_1 = \pi_{\ell r}(t)$ and $d_n = \pi_{\ell r}(t')$ be the path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$ and $d'_1, \ldots, d'_\ell$ the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$. By construction of $\mathcal{I}$ from $\mathcal{J}$, this means that there is a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \sqsubseteq^{*}_{\mathcal{R}} r$ and $(d_i, d_{i+1}) \in s^{\mathcal{J}}$ for all $1 \leq i < n$. Again by construction of $\mathcal{I}$, this means $(d'_i, d'_{i+1}) \in s^{\mathcal{I}}$ for $1 \leq i < \ell$ as required. Hence $\mathcal{I} \models^{\pi_{fr}} s(t_i, t_{i+1})$ for each i with $1 \leq i < \ell$ by definition of π_{fr}.

For (ii): the mapping π_{fr} differs from $\pi_{\ell r}$ only for the newly introduced variables. Furthermore, we only introduced new role conjuncts within a sub-query $\mathsf{subq}(q_{\ell r}, t_r)$ and $\pi_{\ell r}$ is a split match by assumption. By construction, π_{fr} is a split match and we only have to show that π_{fr} is a forest match. Since π_{fr} is a split match, we can do this "tree by tree".

For each $a \in \mathsf{Inds}(\mathcal{A})$, let $T_a = \{w \mid (a, w) \in \mathsf{ran}(\pi_{fr})\}$. We need to construct a mapping f as specified in Definition 4.9, and we start with its root t_r. If $T_a \neq \emptyset$, let $t_r \in \mathsf{Terms}(q)$ be the unique term such that $\pi_{fr}(t_r) = (a, w_r)$ and there is no $t \in \mathsf{Terms}(q)$ such that $\pi_{fr}(t) = (a, w)$ and w is a proper prefix of w_r. Such a term exists since π_{fr} is a split match and it is unique due to Fact (a) above. Define a *trace* to be a sequence $\bar{w} = w_1 \cdots w_n \in T_a^+$ such that

- $w_1 = w_r$;
- for all $1 \leq i < n$, w_i is the longest proper prefix of w_{i+1}.

Since $\mathcal{I}$ is canonical, each $w_i \in T_a$ is in $\mathbb{N}$. It is not hard to see that $T = \{\bar{w} \mid \bar{w} \text{ is a trace}\} \cup \{\varepsilon\}$ is a tree. For a trace $\bar{w} = w_1 \cdots w_n$, let $\mathsf{Tail}(\bar{w}) = w_n$. Define a mapping f that maps each term t with $\pi_{fr}(t) = (a, w) \in T_a$ to the unique trace $\bar{w}_t$ such that $w = \mathsf{Tail}(\bar{w}_t)$. Let $r(t, t') \in q_{fr}$ such that $\pi_{fr}(t), \pi_{fr}(t') \in T_a$. By construction of q_{fr}, this implies that the length of the relevant path from $\pi_{fr}(t)$ to $\pi_{fr}(t')$ is exactly 2. Thus, $f(t)$ and $f(t')$ are neighbours in T and, hence, π_{fr} is a forest match as required. □

Putting everything together, we get the following theorem, which shows that we can use the ground queries in $\mathsf{ground}_{\mathcal{K}}(q)$ and the queries in $\mathsf{tree}_{\mathcal{K}}(q)$ in order to check whether $\mathcal{K}$ entails q, and checking entailment for tree-shaped and ground queries are well understood problems.

Theorem 4.14. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base, q a Boolean conjunctive query, and $\{q_1, \ldots, q_\ell\} = \mathsf{tree}_{\mathcal{K}}(q) \cup \mathsf{ground}_{\mathcal{K}}(q)$. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$.*

Proof. For the "if" direction: let us assume that $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$. Hence, for each model $\mathcal{I}$ of $\mathcal{K}$, there is a query q_i with $1 \leq i \leq \ell$ such that $\mathcal{I} \models q_i$. We distinguish two cases: (i) $q_i \in \mathsf{tree}_{\mathcal{K}}(q)$ and (ii) $q_i \in \mathsf{ground}_{\mathcal{K}}(q)$.

For (i): q_i is of the form $C(v)$ where C is the query concept for some query q_{fr} w.r.t. $v \in \mathsf{Vars}(q_{fr})$ and $(q_{fr}, \emptyset) \in \mathsf{fr}_{\mathcal{K}}(q)$. Hence $\mathcal{I} \models^{\pi} q_i$ for some match π, and thus $\mathcal{I} \models^{\pi} C(v)$. Let $d \in \Delta^{\mathcal{I}}$ with $d = \pi(v) \in C^{\mathcal{I}}$. By Lemma 4.7, we then have that $\mathcal{I} \models q_{fr}$ and, by Lemma 4.13, we then have that $\mathcal{I} \models q$ as required.

For (ii): since $q_i \in \mathsf{ground}_{\mathcal{K}}(q)$, there is some pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ such that $q_i = \mathsf{ground}(q_{fr}, R, \tau)$. We show that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for some match π_{fr}. Since $\mathcal{I} \models q_i$, there is a match π_i such that $\mathcal{I} \models^{\pi_i} q_i$. We now construct the match π_{fr}. For each $t \in R$, q_i contains a concept conjunct $C(\tau(t))$ where $C = \mathsf{con}(\mathsf{subq}(q_{fr}, t), t)$ is the query concept of $\mathsf{subq}(q_{fr}, t)$ w.r.t. t. Since $\mathcal{I} \models^{\pi_i} C(\tau(t))$ and by Lemma 4.7, there is a match π_t such that $\mathcal{I} \models^{\pi_t} \mathsf{subq}(q_{fr}, t)$. We now define π_{fr} as the union of π_t, for each $t \in R$. Please note that $\pi_{fr}(t) = \pi_i(\tau(t))$. Since $\mathsf{Inds}(q_{fr}) \subseteq R$ and τ is such that, for each $a \in \mathsf{Inds}(q_{fr})$, $\tau(a) = a$ and $\tau(t) = \tau(t')$ iff $t \overset{*}{\approx} t'$, it follows that $\mathcal{I} \models^{\pi_{fr}} at$ for each conjunct $at \in q_{fr}$ such that at contains only terms from the root choice R and hence $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ as required.

For the "only if" direction we have to show that, if $\mathcal{K} \models q$, then $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$, so let us assume that $\mathcal{K} \models q$. By Lemma 4.2 in its negated form we have that $\mathcal{K} \models q$ iff all canonical models $\mathcal{I}$ of $\mathcal{K}$ are such that $\mathcal{I} \models q$. Hence, we can restrict our attention to the canonical models of $\mathcal{K}$. By Lemma 4.13, $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \models q$ implies that there is a pair $(q_{fr}, R) \in \mathsf{fr}_\mathcal{K}(q)$ such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for a forest match π_{fr}, R the induced root splitting of π_{fr}, and π_{fr} an injection modulo $\overset{*}{\approx}$. We again distinguish two cases:

(i) $R = \emptyset$, i.e., the root splitting is empty and π_{fr} is a tree match, and

(ii) $R \neq \emptyset$, i.e., the root splitting is non-empty and π_{fr} is a forest match but not a tree match.

For (i): since $(q_{fr}, \emptyset) \in \mathsf{fr}_\mathcal{K}(q)$, there is some $v \in \mathsf{Terms}(q_{fr})$ such that $C = \mathsf{con}(q_{fr}, v)$ and $q_i = C(v)$. By Lemma 4.7 and, since $\mathcal{I} \models q_{fr}$, there is an element $d \in \Delta^\mathcal{I}$ such that $d \in C^\mathcal{I}$. Hence $\mathcal{I} \models^\pi C(v)$ with $\pi : v \mapsto d$ as required.

For (ii): since R is the root splitting induced by π_{fr}, for each $t \in R$ there is some $a_t \in \mathsf{Inds}(\mathcal{A})$ such that $\pi_{fr}(t) = (a_t, \varepsilon)$. We now define the mapping $\tau \colon R \to \mathsf{Inds}(\mathcal{A})$ as follows: for each $t \in R$, $\tau(t) = a_t$ iff $\pi_{fr}(t) = (a_t, \varepsilon)$. By definition of $\mathsf{ground}(q_{fr}, R, \tau)$, $q_i = \mathsf{ground}(q_{fr}, R, \tau) \in \mathsf{ground}_\mathcal{K}(q)$. Since $\mathcal{I} \models^{\pi_{fr}} q_{fr}$, $\mathcal{I} \models \mathsf{subq}(q_{fr}, t)$ for each $t \in R$. Since q_{fr} is forest-shaped, each $\mathsf{subq}(q_{fr}, t)$ is tree-shaped. Then, by Lemma 4.7, $\mathcal{I} \models q_i'$, where q_i' is the query obtained from q_{fr} by replacing each sub-query $\mathsf{subq}(q_{fr}, t)$ with $C(t)$ for C the query concept of $\mathsf{subq}(q_{fr}, t)$ w.r.t. t. By definition of τ from the forest match π_{fr}, it is clear that $\mathcal{I} \models \mathsf{ground}(q_{fr}, R, \tau)$ as required. □

We now give upper bounds on the size and number of queries in $\mathsf{tree}_\mathcal{K}(q)$ and $\mathsf{ground}_\mathcal{K}(q)$. As before, we use $\sharp(S)$ to denote the *cardinality* of a set S. The *size* $|\mathcal{K}|$ ($|q|$) of a knowledge base $\mathcal{K}$ (a query q) is simply the number of symbols needed to write it over the alphabet of constructors, concept names, and role names that occur in $\mathcal{K}$ (q), where numbers in number restrictions are encoded in binary. Obviously, the number of conjuncts in a query is bounded by its size, hence $\sharp(q) \leq |q|$ and, for simplicity, we use n as the size and the cardinality of q in what follows.

Lemma 4.15. *Let q be a Boolean conjunctive query, $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ a $\mathcal{SHIQ}$ knowledge base, $|q| = n$ and $|\mathcal{K}| = m$. Then there is a polynomial p such that*

1. *$\sharp(\mathsf{co}(q)) \leq 2^{p(n)}$ and, for each $q' \in \mathsf{co}(q)$, $|q'| \leq p(n)$,*

2. $\sharp(\mathsf{sr}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot\log p(m)}$, *and, for each* $q' \in \mathsf{sr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,

3. $\sharp(\mathsf{lr}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot\log p(m)}$, *and, for each* $q' \in \mathsf{lr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,

4. $\sharp(\mathsf{fr}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot\log p(m)}$, *and, for each* $q' \in \mathsf{fr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,

5. $\sharp(\mathsf{tree}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot\log p(m)}$, *and, for each* $q' \in \mathsf{tree}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$, *and*

6. $\sharp(\mathsf{ground}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot\log p(m)}$, *and, for each* $q' \in \mathsf{ground}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$.

Proof.

1. The set $\mathsf{co}(q)$ contains those queries obtained from q by adding at most n equality conjuncts to q. The number of collapsings corresponds, therefore, to building all equivalence classes over the terms in q by $\overset{*}{\approx}$. Hence, the cardinality of the set $\mathsf{co}(q)$ is at most exponential in n. Since we add at most one equality conjunct for each pair of terms, the size of a query $q' \in \mathsf{co}(q)$ is at most $n + n^2$, and $|q'|$ is, therefore, polynomial in n.

2. For each of the at most n role conjuncts, we can choose to do nothing, replace the conjunct with two conjuncts, or with three conjuncts. For every replacement, we can choose to introduce a new variable or re-use one of the existing variables. If we introduce a new variable every time, the new query contains at most $3n$ terms. Since $\mathcal{K}$ can contain at most m non-simple roles that are a sub-role of a role used in role conjuncts of q, we have at most m roles to choose from when replacing a role conjunct. Overall, this gives us at most $1 + m(3n) + m(3n)(3n)$ choices for each of the at most n role conjuncts in a query and, therefore, the number of split rewritings for each query $q' \in \mathsf{co}(q)$ is polynomial in m and exponential in n. In combination with the results from (1), this also shows that the overall number of split rewritings is polynomial in m and exponential in n.

   Since we add at most two new role conjuncts for each of the existing role conjuncts, the size of a query $q' \in \mathsf{sr}_{\mathcal{K}}(q)$ is linear in n.

3. There are at most n role conjuncts of the form $r(t, t)$ in a query $q' \in \mathsf{sr}_{\mathcal{K}}(q)$ that could give rise to a loop rewriting, at most m non-simple sub-roles of r in $\mathcal{K}$ that can be used in the loop rewriting, and we can introduce at most one new variable for each role conjunct $r(t, t)$. Therefore, for each query in $\mathsf{sr}_{\mathcal{K}}(q)$, the number of loop rewritings is again polynomial in m

and exponential in n. Combined with the results from (2), this bound also holds for the cardinality of $\mathsf{lr}_{\mathcal{K}}(q)$.

In a loop rewriting, one role conjunct is replaced with two role conjuncts, hence, the size of a query $q' \in \mathsf{lr}_{\mathcal{K}}(q)$ at most doubles.

4. We can use similar arguments as above in order to derive a bound that is exponential in n and polynomial in m for the number of forest rewritings in $\mathsf{fr}_{\mathcal{K}}(q)$.

   Since the number of role conjuncts that we can introduce in a forest rewriting is polynomial in n, the size of each query $q' \in \mathsf{fr}_{\mathcal{K}}(q)$ is at most quadratic in n.

5. The cardinality of the set $\mathsf{tree}_{\mathcal{K}}(q)$ is clearly also polynomial in m and exponential in n since each query in $\mathsf{fr}_{\mathcal{K}}(q)$ can contribute at most one query to the set $\mathsf{tree}_{\mathcal{K}}(q)$. It is not hard to see that the size of a query $q' \in \mathsf{tree}_{\mathcal{K}}(q)$ is polynomial in n.

6. By (1)-(4) above, the number of terms in a root splitting is polynomial in n and there are at most m individual names occurring in $\mathcal{A}$ that can be used for the mapping τ from terms to individual names. Hence the number of different ground mappings τ is at most polynomial in m and exponential in n. The number of ground queries that a single tuple $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ can contribute is, therefore, also at most polynomial in m and exponential in n. Together with the bound on the number of forest rewritings from (4), this shows that the cardinality of $\mathsf{ground}_{\mathcal{K}}(q)$ is polynomial in m and exponential in n. Again it is not hard to see that the size of each query $q' \in \mathsf{ground}_{\mathcal{K}}(q)$ is polynomial in n.

□

As a consequence of the above lemma, there is a bound on the number of queries in $\mathsf{ground}_{\mathcal{K}}(q)$ and $\mathsf{tree}_{\mathcal{K}}(q)$ and it is not hard to see that the two sets can be computed in time polynomial in m and exponential in n. At this point, any decision procedure for $\mathcal{SHIQ}^{\sqcap}$ knowledge base consistency can be used for deciding query entailment. In the following section, we present one such decision procedure that is based on an extension of the translation to looping tree automata given by Tobies [132].

4.3 Deciding Query Entailment for $\mathcal{SHIQ}$

We now devise a decision procedure for entailment of unions of Boolean conjunctive queries that uses, for each disjunct, the queries obtained in the rewriting process as defined in the previous section. For a knowledge base $\mathcal{K}$ and a union of Boolean conjunctive queries $q_1 \vee \ldots \vee q_\ell$, we show how we can use the queries in $\mathsf{tree}_{\mathcal{K}}(q_i)$ and $\mathsf{ground}_{\mathcal{K}}(q_i)$ for $1 \leq i \leq \ell$ in order to build a set of knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n$ such that $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$ iff all the $\mathcal{K}_i$ are inconsistent. This gives rise to two decision procedures: a deterministic one in which we enumerate all $\mathcal{K}_i$, and which we use to derive a tight upper bound for the combined complexity; and a non-deterministic one in which we guess a $\mathcal{K}_i$, and which yields a tight upper bound for the data complexity. Recall that, for combined complexity, the knowledge base $\mathcal{K}$ and the queries q_i both count as input, whereas for the data complexity only the ABox $\mathcal{A}$ counts as an input, and all other parts are assumed to be fixed.

4.3.1 A Deterministic Decision Procedure

We first define the deterministic version of the decision procedure and give an upper bound for its combined complexity. The given algorithm takes as input a union of connected conjunctive queries and works under the unique name assumption (UNA). We show afterwards how it can be extended to an algorithm that does not make the UNA and that takes arbitrary UCQs as input, and that the complexity results carry over.

We now construct a set of knowledge bases that extend the original knowledge base $\mathcal{K}$ both w.r.t. the TBox and ABox. The extended knowledge bases are such that a given KB $\mathcal{K}$ entails a query q iff all the extended KBs are inconsistent. We handle the concepts obtained from the tree-shaped queries differently to the ground queries: the axioms we add to the TBox prevent matches for the tree-shaped queries, whereas the extended ABoxes contain assertions that prevent matches for the ground queries.

Definition 4.16. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and $q = q_1 \vee \ldots \vee q_\ell$ a union of Boolean conjunctive queries. We set

1. $T = \mathsf{tree}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{tree}_{\mathcal{K}}(q_\ell)$,
2. $G = \mathsf{ground}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{ground}_{\mathcal{K}}(q_\ell)$, and

3. $\mathcal{T}_q = \{\top \sqsubseteq \neg C \mid C(v) \in T\}$.

An *extended knowledge base* $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and q is a $\mathcal{SHIQ}^{\sqcap}$ knowledge base $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$, where $\mathcal{A}_q = \{\neg co \mid co \in A_q\}$ and A_q is a set of conjuncts such that, for each ground query $q' \in G$, A_q contains at least one conjunct from q'. △

Informally, the extended TBox $\mathcal{T} \cup \mathcal{T}_q$ ensures that there are no tree matches. Each extended ABox $\mathcal{A} \cup \mathcal{A}_q$ contains, for each ground query q' obtained in the rewriting process, at least one assertion $\neg co$ with $co \in q'$ that "spoils" a match for q'. A model for such an extended ABox can, therefore, not satisfy any of the ground queries. If there is a model for any of the extended knowledge bases, we know that this is a counter-model for the original query.

We can now use the extended knowledge bases in order to define the deterministic version of our algorithm for deciding entailment of unions of Boolean conjunctive queries in $\mathcal{SHIQ}$.

Algorithm 4.1 SHIQ_UCQ_Entailment_Deterministic

Input:
$\mathcal{K}$: a $\mathcal{SHIQ}$ knowledge base
q: a union of connected Boolean conjunctive queries

Output:
true: if $\mathcal{K} \models q$
false: if $\mathcal{K} \not\models q$

```
1: for each extended knowledge base K_q w.r.t. K and q do
2:     if K_q is satisfiable then
3:         return false
4:     end if
5: end for
6: return true
```

The following lemma shows that the above described algorithm is indeed correct.

Lemma 4.17. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and q a union of connected Boolean conjunctive queries. Given $\mathcal{K}$ and q as input, Algorithm 4.3.1 answers* true *iff $\mathcal{K} \models q$ under the unique name assumption.*

Proof. For the "only if"-direction: let $q = q_1 \vee \ldots \vee q_\ell$. We show the contrapositive and assume that $\mathcal{K} \not\models q$. We can assume that $\mathcal{K}$ is consistent since an inconsistent

knowledge base trivially entails every query. Let $\mathcal{I}$ be a model of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. We show that $\mathcal{I}$ is also a model of some extended knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$. We first show that $\mathcal{I}$ is a model of $\mathcal{T}_q$. To this end, let $\top \sqsubseteq \neg C$ in $\mathcal{T}_q$. Then $C(v) \in T$ with $C = \mathsf{con}(q_{fr}, v)$ for some pair $(q_{fr}, \emptyset) \in \mathsf{fr}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{fr}_\mathcal{K}(q_\ell)$ and $v \in \mathsf{Vars}(q_{fr})$. Let i be such that $(q_{fr}, \emptyset) \in \mathsf{fr}_\mathcal{K}(q_i)$. Now $C^\mathcal{I} \neq \emptyset$ implies, by Lemma 4.7, that $\mathcal{I} \models q_{fr}$ and, by Lemma 4.13, $\mathcal{I} \models q_i$ and, hence, $\mathcal{I} \models q$, contradicting our assumption. Thus $\mathcal{I} \models \top \sqsubseteq \neg C$ for each $\top \sqsubseteq \neg C \in \mathcal{T}_q$ and, thus, $\mathcal{I} \models \mathcal{T}_q$.

Next, we define an extended ABox $\mathcal{A}_q$ such that, for each $q' \in G$,

- if $C(a) \in q'$ and $a^\mathcal{I} \in \neg C^\mathcal{I}$, then $\neg C(a) \in \mathcal{A}_q$;
- if $r(a, b) \in q'$ and $(a^\mathcal{I}, b^\mathcal{I}) \notin r^\mathcal{I}$, then $\neg r(a, b) \in \mathcal{A}_q$.

Now assume that we have a query $q' = \mathsf{ground}(q_{fr}, R, \tau) \in \mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$ such that there is no conjunct $co \in q'$ with $\neg co \in \mathcal{A}_q$. Then trivially $\mathcal{I} \models q'$. Let i be such that $(q_{fr}, R) \in \mathsf{fr}_\mathcal{K}(q_i)$. By Theorem 4.14, $\mathcal{I} \models q_i$ and thus $\mathcal{I} \models q$, which is a contradiction to our assumption. Hence $\mathcal{K}_q$ is an extended knowledge base and $\mathcal{I} \models \mathcal{K}_q$ as required.

For the "if"-direction, we assume that $\mathcal{K} \models q$, but the algorithm answers "$\mathcal{K}$ does not entail q". Hence there is an extended knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ that is consistent, i.e., there is a model $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}_q$. Since $\mathcal{K}_q$ is an extension of $\mathcal{K}$, $\mathcal{I} \models \mathcal{K}$. Moreover, we have that $\mathcal{I} \models \mathcal{T}_q$ and, hence, for each $d \in \Delta^\mathcal{I}$, $d \in \neg C^\mathcal{I}$ for each $C(v) \in \mathsf{tree}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{tree}_\mathcal{K}(q_\ell)$. By Lemma 4.7, we then have that $\mathcal{I} \not\models q'$ for each $q' \in \mathsf{tree}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{tree}_\mathcal{K}(q_\ell)$ and, by Lemma 4.13, $\mathcal{I} \not\models q_i$ for each i with $1 \leq i \leq \ell$.

By definition of extended knowledge bases, $\mathcal{A}_q$ contains an assertion $\neg co$ for at least one conjunct co in each query $q' = \mathsf{ground}(q_{fr}, R, \tau)$ from $\mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$. Hence $\mathcal{I} \not\models q'$ for each $q' \in \mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$. Then, by Theorem 4.14, $\mathcal{I} \not\models q$, which contradicts our assumption. □

Combined Complexity of Query Entailment in $\mathcal{SHIQ}$

According to the above lemma, Algorithm 4.3.1 is correct. We now analyse its combined complexity and thereby prove that it is also terminating.

As mentioned in the preliminaries (cf. 2.4), we assume for the complexity analysis that all concepts in ABox assertions and concept conjuncts of the input query use only literals, i.e., concept names or negated concept names. This is

without loss of generality and has the advantage that the size of each conjunct and ABox assertion is constant.

Since our algorithm involves checking the consistency of a $\mathcal{SHIQ}^{\sqcap}$ knowledge base, we now analyse the complexity of this reasoning service. Tobies [132] proves an ExpTime upper bound for deciding the consistency of $\mathcal{SHIQ}$ knowledge bases (even with binary coding of numbers) by translating a $\mathcal{SHIQ}$ KB to an equisatisfiable $\mathcal{ALCQI}b$ knowledge base. The b stands for *safe Boolean role expressions* built from $\mathcal{ALCQI}b$ roles using the operator $\sqcap$ (role intersection), $\sqcup$ (role union), and $\neg$ (role negation/complement) such that, when transformed into disjunctive normal form, every disjunct contains at least one non-negated conjunct. As for $\mathcal{SHIQ}$, checking the consistency of an $\mathcal{ALCQI}b$ knowledge base is an ExpTime-complete problem even with binary coding of numbers in number restrictions.

Given a query q and a $\mathcal{SHIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, we reduce query entailment to deciding knowledge base consistency of an extended $\mathcal{SHIQ}^{\sqcap}$ knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$. Recall that $\mathcal{T}_q$ and $\mathcal{A}_q$ are the only parts that contain role conjunctions and that we use role negation only in ABox assertions. We extend the translation given for $\mathcal{SHIQ}$ so that it can be used for deciding the consistency of $\mathcal{SHIQ}^{\sqcap}$ KBs. Although the translation works for all $\mathcal{SHIQ}^{\sqcap}$ KBs, we assume in our complexity analysis that the input KB is of exactly the form of extended knowledge bases as described above. This is so because the translation for unrestricted $\mathcal{SHIQ}^{\sqcap}$ is no longer polynomial, as in the case of $\mathcal{SHIQ}$, but exponential in the size of the longest role conjunction under a universal quantifier. In extended knowledge bases role conjunctions occur by definition only in the extended ABox and TBox. Since, by Lemma 4.15, the size of each role conjunction in an extended knowledge base is polynomial in the size of q, the translation is only exponential in the size of the query in the case of extended knowledge bases.

We define the *closure* $\mathsf{cl}(C, \mathcal{R})$ of a $\mathcal{SHIQ}$-concept C w.r.t. a role hierarchy $\mathcal{R}$ as the smallest set satisfying the following conditions:

- if D is a sub-concept of C, then $D \in \mathsf{cl}(C, \mathcal{R})$,
- if $D \in \mathsf{cl}(C, \mathcal{R})$, then $\mathsf{nnf}(\neg D) \in \mathsf{cl}(C, \mathcal{R})$,
- if $\forall r.D \in \mathsf{cl}(C, \mathcal{R})$, $s \sqsubseteq^{*}_{\mathcal{R}} r$, and $s \in \mathsf{Trans}_{\mathcal{R}}$, then $\forall s.D \in \mathsf{cl}(C, \mathcal{R})$.

We now show how we can extend the translation from $\mathcal{SHIQ}$ into $\mathcal{ALCQIb}$ given by Tobies to $\mathcal{SHIQ}^{\sqcap}$. We first consider $\mathcal{SHIQ}^{\sqcap}$-concepts and then extend the translation to knowledge bases.

Definition 4.18. For a role hierarchy $\mathcal{R}$ and roles $r, r_1, \ldots, r_n$, let

$$\uparrow(r, \mathcal{R}) = \bigsqcap_{r \sqsubseteq^*_{\mathcal{R}} s} s \quad \text{and} \quad \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}) = \uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}).$$

△

When we apply the $\uparrow$ operator to a role r, we conjoin to r all super-roles that are implied by the given role hierarchy. Since this, intuitively, explicates all implied roles, we can afterwards discard the role hierarchy. When we say that we close a role upwards, we mean that we apply the $\uparrow$ operator to the role. Please note also that, since $r \sqsubseteq^*_{\mathcal{R}} r$, r occurs in $\uparrow(r, \mathcal{R})$.

Lemma 4.19. *Let $\mathcal{R}$ be a role hierarchy, and $r_1, \ldots, r_n$ roles. For every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}$, it holds that $(\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}))^{\mathcal{I}} = (r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}}$.*

Proof. The proof is a straightforward extension of Lemma 6.19 in [132]. By definition, $\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}) = \uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R})$ and, by definition of the semantics of role conjunctions, we have that $(\uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}))^{\mathcal{I}} = \uparrow(r_1, \mathcal{R})^{\mathcal{I}} \cap \ldots \cap \uparrow(r_n, \mathcal{R})^{\mathcal{I}}$. If $s \sqsubseteq^*_{\mathcal{R}} r$, then $\{s' \mid r \sqsubseteq^*_{\mathcal{R}} s'\} \subseteq \{s' \mid s \sqsubseteq^*_{\mathcal{R}} s'\}$ and hence $\uparrow(s, \mathcal{R})^{\mathcal{I}} \subseteq \uparrow(r, \mathcal{R})^{\mathcal{I}}$. If $\mathcal{I} \models \mathcal{R}$, then $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for every s with $r \sqsubseteq^*_{\mathcal{R}} s$. Hence, $\uparrow(r, \mathcal{R})^{\mathcal{I}} = r^{\mathcal{I}}$ and $(\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}))^{\mathcal{I}} = (\uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}))^{\mathcal{I}} = \uparrow(r_1, \mathcal{R})^{\mathcal{I}} \cap \ldots \cap \uparrow(r_n, \mathcal{R})^{\mathcal{I}} = {r_1}^{\mathcal{I}} \cap \ldots \cap {r_n}^{\mathcal{I}} = (r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}}$ as required. □

With the extended definition of $\uparrow$ on role conjunctions, we can now adapt the definition (Definition 6.22 in [132]) that Tobies provides for translating $\mathcal{SHIQ}$-concepts into $\mathcal{ALCQIb}$-concepts.

Definition 4.20. Let C be a $\mathcal{SHIQ}^{\sqcap}$-concept in NNF and $\mathcal{R}$ a role hierarchy. For every concept $\forall(r_1 \sqcap \ldots \sqcap r_n).D \in \mathsf{cl}(C, \mathcal{R})$, let $X_{r_1 \sqcap \ldots \sqcap r_n, D} \in N_C$ be a unique

concept name that does not occur in $\mathsf{cl}(C, \mathcal{R})$. We define the function tr inductively on the structure of concepts by setting

$$\begin{aligned}
\mathsf{tr}(A, \mathcal{R}) &= A \text{ for all A } \in N_C \\
\mathsf{tr}(\neg A, \mathcal{R}) &= \neg A \text{ for all A } \in N_C \\
\mathsf{tr}(C_1 \sqcap C_2, \mathcal{R}) &= \mathsf{tr}(C_1, \mathcal{R}) \sqcap \mathsf{tr}(C_2, \mathcal{R}) \\
\mathsf{tr}(C_1 \sqcup C_2, \mathcal{R}) &= \mathsf{tr}(C_1, \mathcal{R}) \sqcup \mathsf{tr}(C_2, \mathcal{R}) \\
\mathsf{tr}(\bowtie n(r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) &= (\bowtie n \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}).\mathsf{tr}(D, \mathcal{R})) \\
\mathsf{tr}(\forall(r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) &= X_{r_1 \sqcap \ldots \sqcap r_n, D} \\
\mathsf{tr}(\exists(r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) &= \neg(X_{r_1 \sqcap \ldots \sqcap r_n, \mathsf{nnf}(\neg D)})
\end{aligned}$$

where $\bowtie$ stands for $\leqslant$ or $\geqslant$. For a conjunction of roles $r_1 \sqcap \ldots \sqcap r_n$, set $\mathsf{tc}((r_1 \sqcap \ldots \sqcap r_n), \mathcal{R}) = \{(t_1 \sqcap \ldots \sqcap t_n) \mid t_i \sqsubseteq^*_{\mathcal{R}} r_i \text{ and } t_i \in \mathsf{Trans}_{\mathcal{R}} \text{ for each } i \text{ such that } 1 \leq i \leq n\}$ and define an extended TBox $\mathcal{T}_{C,\mathcal{R}}$ as

$$\begin{aligned}
&\mathcal{T}_{C,\mathcal{R}} = \\
&\{X_{r_1 \sqcap \ldots \sqcap r_n, D} \equiv \forall \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}).\mathsf{tr}(D, \mathcal{R}) \mid \forall(r_1 \sqcap \ldots \sqcap r_n).D \in \mathsf{cl}(C, \mathcal{R})\} \cup \\
&\{X_{r_1 \sqcap \ldots \sqcap r_n, D} \sqsubseteq \forall \uparrow(T, \mathcal{R}).X_{T,D} \mid T \in \mathsf{tc}(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R})\}
\end{aligned}$$

△

Informally, the axioms in the TBox $\mathcal{T}_{C,\mathcal{R}}$ encode the transitivity of roles and make sure that concepts under universal quantification are propagated accordingly. Similar encodings are known for various Modal and Description Logics [74, 117] or are captured by the $\forall^+$-rule in many tableau algorithms [63, 68]. Given, additionally, the results from Lemma 4.19, which shows that upwards closing the roles in concepts makes the role hierarchy redundant, intuitively justifies the following lemma.

Lemma 4.21. *Let C be a $\mathcal{SHIQ}^{\sqcap}$-concept in NNF, $\mathcal{R}$ a role hierarchy, and tr and $\mathcal{T}_{C,\mathcal{R}}$ as defined in Definition 4.20. The concept C is satisfiable w.r.t. $\mathcal{R}$ iff the $\mathcal{ALCQIb}$-concept $\mathsf{tr}(C, \mathcal{R})$ is satisfiable w.r.t. $\mathcal{T}_{C,\mathcal{R}}$.*

Given the extension of $\uparrow(r, \mathcal{R})$ to role conjunctions in Definition 4.18 and the proof of Lemma 4.19, which shows that we can capture the semantics of a given role hierarchy by using role conjunctions, the detailed proof of Lemma 4.21 given by Tobies 2001, straightforwardly extends to our case.

We now analyse the complexity of the above described problem. Let $m = |\mathcal{R}|$ and $r_1 \sqcap \ldots \sqcap r_n$ the longest role conjunction occurring in C, i.e., the maximal number of roles that occur in a role conjunction in C is n. The TBox $\mathcal{T}_{C,\mathcal{R}}$ can

contain exponentially many axioms in n since the cardinality of the set $\mathsf{tc}((r_1 \sqcap \ldots \sqcap r_n), \mathcal{R})$ for the longest role conjunction can only be bounded by m^n because each r_i can have up tp m transitive sub-roles. It is not hard to show that the size of each axiom is polynomial in $|C|$. Since deciding whether an $\mathcal{ALCQIb}$ concept C is satisfiable w.r.t. an $\mathcal{ALCQIb}$ TBox $\mathcal{T}$ is an ExpTime-complete problem (even with binary coding of numbers), the satisfiability of a $\mathcal{SHIQ}^{\sqcap}$-concept C can be checked in time $2^{p(m)2^{p(n)}}$ for some polynomial p.

We now extend the translation from concepts to knowledge bases. Tobies assumes that all role assertions in the ABox are of the form $r(a, b)$ with r a role name or the inverse of a role name. Extended ABoxes contain, however, also negated roles in role assertions, which require a different translation. A positive role assertion such as $r(a, b)$ is translated in the standard way by closing the role upwards. The only difference of using $\uparrow$ directly is that we additionally split the conjunction $(\uparrow(r, \mathcal{R}))(a, b) = (r_1 \sqcap \ldots \sqcap r_n)(a, b)$ into n different role assertions $r_1(a, b), \ldots, r_n(a, b)$, which is clearly justified by the semantics. For negated roles in a role assertion such as $\neg r(a, b)$, we close the role downwards instead of upwards, which means that we add a role conjunct $\neg s(a, b)$ for each sub-role s of r. This is again justified by the semantics because each sub-role s of r would immediately imply $r(a, b)$, which contradict the assertion $\neg r(a, b)$. After closing the negated roles in the ABox downwards, we can again discard the initial role hierarchy.

Let $\mathcal{K} = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ be an extended knowledge base. More precisely, we set

$$\mathsf{tr}(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}) = \{\mathsf{tr}(C, \mathcal{R}) \sqsubseteq \mathsf{tr}(D, \mathcal{R}) \mid C \sqsubseteq D \in \mathcal{T} \cup \mathcal{T}_q\},$$

$$\begin{aligned}\mathsf{tr}(\mathcal{A} \cup \mathcal{A}_q, \mathcal{R}) = \; & \{(\mathsf{tr}(C, \mathcal{R}))(a) \mid C(a) \in \mathcal{A} \cup \mathcal{A}_q\} \cup \\ & \{s(a, b) \mid r(a, b) \in \mathcal{A} \cup \mathcal{A}_q \text{ and } r \sqsubseteq^*_{\mathcal{R}} s\} \cup \\ & \{\neg s(a, b) \mid \neg r(a, b) \in \mathcal{A} \cup \mathcal{A}_q \text{ and } s \sqsubseteq^*_{\mathcal{R}} r\},\end{aligned}$$

and we use $\mathsf{tr}(\mathcal{K}, \mathcal{R})$ to denote the $\mathcal{ALCQIb}$ knowledge base $(\mathsf{tr}(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}), \mathsf{tr}(\mathcal{A} \cup \mathcal{A}_q, \mathcal{R}))$.

For the complexity of deciding the consistency of a translated $\mathcal{SHIQ}^{\sqcap}$ knowledge base, we can apply the same arguments as above for concept satisfiability, which gives the following result:

Lemma 4.22. *Given a $\mathcal{SHIQ}^{\sqcap}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $m = |\mathcal{K}|$ and the size of the longest role conjunction is n, we can decide consistency of $\mathcal{K}$*

in deterministic time $2^{p(m)2^{p(n)}}$ *with* p *a polynomial.*

Proof. We first translate $\mathcal{K}$ into the $\mathcal{ALCQI}b$ knowledge base

$$\mathsf{tr}(\mathcal{K}, \mathcal{R}) = (\mathsf{tr}(\mathcal{T}, \mathcal{R}), \mathsf{tr}(\mathcal{A}, \mathcal{R})).$$

Since the longest role conjunction is of size n, the cardinality of each set $\mathsf{tc}(R, \mathcal{R})$ for a role conjunction R is bounded by m^n. Hence, the TBox $\mathsf{tr}(\mathcal{T}, \mathcal{R})$ can contain exponentially many axioms in n. It is not hard to check that the size of each axiom is polynomial in m. Since deciding whether an $\mathcal{ALCQI}b$ KB is consistent is an ExpTime-complete problem (even with binary coding of numbers) [132, Theorem 4.42], the consistency of $\mathsf{tr}(\mathcal{K}, \mathcal{R})$ can be checked in time $2^{p(m)2^{p(n)}}$. □

We are now ready to show that Algorithm 4.3.1 runs in deterministic time single exponential in the size of the input KB and double exponential in the size of the input query.

Lemma 4.23. *Let* $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ *be a* $\mathcal{SHIQ}$ *knowledge base with* $m = |\mathcal{K}|$ *and* q *a union of connected Boolean conjunctive queries with* $n = |q|$*. Given* $\mathcal{K}$ *and* q *as input, Algorithm 4.3.1 decides whether* $\mathcal{K} \models q$ *under the unique name assumption in deterministic time in* $2^{p(m)2^{p(n)}}$ *for some polynomial* p*.*

Proof. We first show that there is some polynomial p such that we have to check at most $2^{p(m)2^{p(n)}}$ extended knowledge bases for consistency and then that each consistency check can be done in time $2^{p(m)2^{p(n)}}$, which gives an upper bound of $2^{p(m)2^{p(n)}}$ on the time needed for deciding whether $\mathcal{K} \models q$.

Let $q = q_1 \vee \ldots \vee q_\ell$. Clearly, we can use n as a bound for ℓ, i.e., $\ell \leq n$. Moreover, the size of each query q_i with $1 \leq i \leq \ell$ is bounded by n. Together with Lemma 4.15, we get that $\sharp(T)$ and $\sharp(G)$ are bounded by $2^{p(n)\cdot \log p(m)}$ for some polynomial p and it is clear that these sets can be computed in this time bound as well.

Due to Lemma 4.15 (5), the size of each query $q' \in T$ is polynomial in n. Computing a query concept $C_{q'}$ of q' w.r.t. some variable $v \in \mathsf{Vars}(q')$ can be done in time polynomial in n. Thus the TBox $\mathcal{T}_q$ can be computed in time $2^{p(n)\cdot \log p(m)}$.

The size of each query $q' \in G$ w.r.t. an ABox $\mathcal{A}$ is polynomial in n and, when constructing $\mathcal{A}_q$, we can add a subset of (negated) conjuncts from each $q' \in G$ to $\mathcal{A}_q$. Hence, there are at most $2^{p(m)2^{p(n)}}$ extended ABoxes $\mathcal{A}_q$ and, therefore,

$2^{p(m)2^{p(n)}}$ extended knowledge bases that have to be tested for consistency. The size of an extended ABox $\mathcal{A}_q$ is maximal if we add, for each of the at most $2^{p(n)\cdot\log p(m)}$ ground queries in G, all conjuncts in their negated form. Since, by Lemma 4.15 (6), the size of these queries is polynomial in n, the size of each extended ABox $\mathcal{A}_q$ is bounded by $2^{p(n)\cdot\log p(m)}$ and it is clear that we can compute an extended ABox in this time bound as well.

Hence, the size of each extended KB $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ is bounded by $2^{p(n)\cdot\log p(m)}$. Since role conjunctions occur only in $\mathcal{T}_q$ or $\mathcal{A}_q$, and the size of each concept in $\mathcal{T}_q$ and $\mathcal{A}_q$ is polynomial in n, the length of the longest role conjunction is also polynomial in n. When translating an extended knowledge base into an $\mathcal{ALCQI}b$ knowledge base, the number of axioms resulting from each concept C that occurs in $\mathcal{T}_q$ or $\mathcal{A}_q$ can be exponential in n. Thus, the size of each resulting $\mathcal{ALCQI}b$ knowledge base is still exponential in n an polynomial in m.

Since deciding whether an $\mathcal{ALCQI}b$ knowledge base is consistent is an ExpTime-complete problem (even with binary coding of numbers) [132, Theorem 4.42], it can be checked in time $2^{p(m)2^{p(n)}}$ whether $\mathcal{K}$ is consistent.

Since we have to check at most $2^{p(m)2^{p(n)}}$ knowledge bases for consistency, and each check can be done in time $2^{p(m)2^{p(n)}}$, we obtain the desired upper bound of $2^{p(m)2^{p(n)}}$ for deciding whether $\mathcal{K} \models q$. □

We now show that this result carries over even when we do not restrict interpretations to the unique name assumption.

Definition 4.24. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and q a $\mathcal{SHIQ}$ union of Boolean conjunctive queries. For a partition $\mathcal{P}$ of $\mathsf{Inds}(\mathcal{A})$, a knowledge base $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and a query $q^{\mathcal{P}}$ are called an *$\mathcal{A}$-partition w.r.t. $\mathcal{K}$ and q* if $\mathcal{A}^{\mathcal{P}}$ and $q^{\mathcal{P}}$ are obtained from $\mathcal{A}$ and q as follows: For each $P \in \mathcal{P}$

1. Choose one individual name $a \in P$.
2. For each $b \in P$, replace each occurrence of b in $\mathcal{A}$ and q with a.

△

Please note that, without loss of generality, we assume that all constants that occur in the query occur in the ABox as well and that thus a partition of the individual names in the ABox also partitions the query.

Lemma 4.25. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and q a union of Boolean conjunctive queries. $\mathcal{K} \not\models q$ without making the unique name assumption iff there is an $\mathcal{A}$-partition $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ w.r.t. $\mathcal{K}$ and q such that $\mathcal{K}^{\mathcal{P}} \not\models q^{\mathcal{P}}$ under the unique name assumption.*

Proof. For the "only if"-direction: Since $\mathcal{K} \not\models q$, there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. Let $f\colon \mathsf{Inds}(\mathcal{A}) \to \mathsf{Inds}(\mathcal{A})$ be a total function such that, for each maximal set of individual names $\{a_1, \ldots, a_n\}$ for which ${a_1}^{\mathcal{I}} = {a_i}^{\mathcal{I}}$ for $1 \leq i \leq n$, $f(a_i) = a_1$. Let $\mathcal{A}^{\mathcal{P}}$ and $q^{\mathcal{P}}$ be obtained from $\mathcal{A}$ and q by replacing each individual name a in $\mathcal{A}$ and q with $f(a)$. Clearly, $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ are an $\mathcal{A}$-partition w.r.t. $\mathcal{K}$ and q. Let $\mathcal{I}^{\mathcal{P}} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{\mathcal{P}}})$ be an interpretation that is obtained by restricting $\cdot^{\mathcal{I}}$ to individual names in $\mathsf{Inds}(\mathcal{A}^{\mathcal{P}})$. It is easy to see that $\mathcal{I}^{\mathcal{P}} \models \mathcal{K}^{\mathcal{P}}$ and that the unique name assumption holds in $\mathcal{I}^{\mathcal{P}}$. We now show that $\mathcal{I}^{\mathcal{P}} \not\models q^{\mathcal{P}}$. Assume, to the contrary of what is to be shown, that $\mathcal{I}^{\mathcal{P}} \models^{\pi'} q^{\mathcal{P}}$ for some match π'. We define a mapping $\pi\colon \mathsf{Terms}(q) \to \Delta^{\mathcal{I}}$ from π' by setting $\pi(a) = \pi'(f(a))$ for each individual name $a \in \mathsf{Inds}(q)$ and $\pi(v) = \pi'(v)$ for each variable $v \in \mathsf{Vars}(q)$. It is easy to see that $\mathcal{I} \models^{\pi} q$, which is a contradiction.

For the "if"-direction: Let $\mathcal{I}^{\mathcal{P}} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{\mathcal{P}}})$ be such that $\mathcal{I}^{\mathcal{P}} \models \mathcal{K}^{\mathcal{P}}$ under the UNA and $\mathcal{I}^{\mathcal{P}} \not\models q^{\mathcal{P}}$ and let $f\colon \mathsf{Inds}(\mathcal{A}) \to \mathsf{Inds}(\mathcal{A}^{\mathcal{P}})$ be a total function such that $f(a)$ is the individual that replaced a in $\mathcal{A}^{\mathcal{P}}$ and $q^{\mathcal{P}}$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that extends $\mathcal{I}^{\mathcal{P}}$ such that $a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}}$. We show that $\mathcal{I} \models \mathcal{K}$ and that $\mathcal{I} \not\models q$. It is clear that $\mathcal{I} \models \mathcal{T}$. Let $C(a)$ be an assertion in $\mathcal{A}$ such that a was replaced with $a_{\mathcal{P}}$ in $\mathcal{A}^{\mathcal{P}}$. Since $\mathcal{I}^{\mathcal{P}} \models C(a_{\mathcal{P}})$ and $a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}} = {a_{\mathcal{P}}}^{\mathcal{I}^{\mathcal{P}}} \in C^{\mathcal{I}^{\mathcal{P}}}$, we also have that $\mathcal{I} \models C(a)$. We can use a similar argument for (possibly negated) role assertions. Let $a \not\doteq b$ be an assertion in $\mathcal{A}$ such that a was replaced with $a_{\mathcal{P}}$ and b with $b_{\mathcal{P}}$ in $\mathcal{A}^{\mathcal{P}}$, i.e., $f(a) = a_{\mathcal{P}}$ and $f(b) = b_{\mathcal{P}}$. Since $\mathcal{I}^{\mathcal{P}} \models a_{\mathcal{P}} \not\doteq b_{\mathcal{P}}$, $a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}} = {a_{\mathcal{P}}}^{\mathcal{I}^{\mathcal{P}}} \neq {b_{\mathcal{P}}}^{\mathcal{I}^{\mathcal{P}}} = f(b)^{\mathcal{I}^{\mathcal{P}}} = b^{\mathcal{I}}$ and $\mathcal{I} \models a \not\doteq b$ as required. Therefore, we have that $\mathcal{I} \models \mathcal{K}$ as required.

Assume that $\mathcal{I} \models^{\pi} q$ for a match π. Let $\pi^{\mathcal{P}}\colon \mathsf{Terms}(q^{\mathcal{P}}) \to \Delta^{\mathcal{I}}$ be a mapping such that $\pi^{\mathcal{P}}(v) = \pi(v)$ for $v \in \mathsf{Vars}(q^{\mathcal{P}})$ and $\pi^{\mathcal{P}}(a_{\mathcal{P}}) = \pi(a)$ for $a_{\mathcal{P}} \in \mathsf{Inds}(q^{\mathcal{P}})$ and some a such that $a_{\mathcal{P}} = f(a)$. Let $C(a_{\mathcal{P}}) \in q^{\mathcal{P}}$ be such that $C(a) \in q$ and a was replaced with $a_{\mathcal{P}}$, i.e., $f(a) = a_{\mathcal{P}}$. By assumption, $\pi(a) \in C^{\mathcal{I}}$, but then $\pi(a) = a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}} = {a_{\mathcal{P}}}^{\mathcal{I}^{\mathcal{P}}} = \pi^{\mathcal{P}}(a_{\mathcal{P}}) \in C^{\mathcal{I}^{\mathcal{P}}}$ and $\mathcal{I}^{\mathcal{P}} \models C(a_{\mathcal{P}})$. Similar arguments can be used to show entailment for role and equality conjuncts, which yields the desired contradiction. □

Let $\mathcal{C}$ be the complexity class of deciding whether $\mathcal{K} \models q$ under the unique

name assumption and $n = 2^{|\mathcal{A}|}$. Since the number of partitions for an ABox is at most exponential in the number of individual names that occur in the ABox, the following is a straightforward consequence of the above lemma: for a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ in a Description Logic $\mathcal{DL}$ and a Boolean conjunctive $\mathcal{DL}$ query q, deciding whether $\mathcal{K} \models q$ without making the unique name assumption can be reduced to deciding n problems in $\mathcal{C}$.

In order to extend our algorithm to unions of possibly unconnected Boolean conjunctive queries, we first transform the input query q into an equivalent query q' that is in conjunctive normal form (CNF) (cf. Lemma 2.9 on page 39). We then check entailment for each conjunct q_i, which is now a union of connected Boolean conjunctive queries. The algorithm returns "$\mathcal{K}$ entails q" if each entailment check succeeds and it answers "$\mathcal{K}$ does not entail q" otherwise. By Lemma 2.9 and Lemma 4.17, the algorithm is correct.

Let $\mathcal{K}$ be a knowledge base in a Description Logic $\mathcal{DL}$, q a union of connected Boolean conjunctive $\mathcal{DL}$ queries, and $\mathcal{C}$ a complexity class such that deciding whether $\mathcal{K} \models q$ is in $\mathcal{C}$. Let q' be a union of possibly unconnected Boolean conjunctive queries with $|q'| = n$ and $\mathsf{cnf}(q')$ the CNF of q'. Since the number of conjuncts in $\mathsf{cnf}(q')$ is at most exponential in n, deciding whether $\mathcal{K} \models q'$ can be reduced to deciding 2^n problems in $\mathcal{C}$. This, together with the results from Lemma 4.23, gives the following general result:

Theorem 4.26. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $m = |\mathcal{K}|$ and q a union of Boolean conjunctive queries with $n = |q|$. Deciding whether $\mathcal{K} \models q$ can be done in deterministic time in $2^{p(m)2^{p(n)}}$ for some polynomial p.*

A corresponding lower bound follows from the 2ExpTime-hardness result for $\mathcal{ALCI}$ [88, Thm. 2].

Corollary 4.27. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base with $m = |\mathcal{K}|$ and q a union of Boolean conjunctive queries with $n = |q|$. Deciding whether $\mathcal{K} \models q$ is a 2ExpTime-complete problem.*

Very recently, an automata-based decision procedure for positive existential path queries over $\mathcal{ALCQIb}_{reg}$ knowledge bases has been presented [30]. Positive existential path queries generalise unions of conjunctive queries and since a $\mathcal{SHIQ}$ knowledge base can be polynomially reduced to an $\mathcal{ALCQIb}_{reg}$ knowledge base, the presented algorithm is a decision procedure for entailment of unions of

conjunctive queries in $\mathcal{SHIQ}$ as well. The automata-based technique is also worst-case optimal regarding the combined complexity and it can be considered more elegant than our rewriting algorithm. It is, however, not suited for studying the data complexity of the problem. With the technique presented here, we can also derive tight upper bounds for the data complexity and we present these results in the following section.

4.3.2 A Non-Deterministic Decision Procedure

In order to study the data complexity of query entailment, we devise a non-deterministic decision procedure which provides a tight bound for the complexity of the problem. Actually, the devised algorithm decides non-entailment of queries: it returns "$\mathcal{K}$ does not entail q" if we can guess an extended knowledge base $\mathcal{K}_q$ that is consistent, and it returns "$\mathcal{K}$ entails q" otherwise.

Algorithm 4.2 SHIQ_UCQ_Entailment_Nondeterministic

Input:
$\mathcal{A}$: a $\mathcal{SHIQ}$ ABox
Output:
false: if $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \not\models q$
true: otherwise

1: Let $\mathcal{T}$ be a $\mathcal{SHIQ}$ TBox
2: Le $\mathcal{R}$ be a $\mathcal{SHIQ}$ role hierarchy
3: Let q be a union of Boolean conjunctive queries
4: guess an $\mathcal{A}$-partition $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ and q
5: let $\mathsf{cnf}(q^{\mathcal{P}})$ be the conjunctive normal form of $q^{\mathcal{P}}$
6: choose a conjunct $q_i^{\mathcal{P}}$ from $\mathsf{cnf}(q^{\mathcal{P}})$
7: guess an extended knowledge base $\mathcal{K}_{q_i}^{\mathcal{P}} = (\mathcal{T} \cup \mathcal{T}_{q_i}, \mathcal{R}, \mathcal{A}^{\mathcal{P}} \cup \mathcal{A}_{q_i}^{\mathcal{P}})$ w.r.t. $\mathcal{K}^{\mathcal{P}}$ and $q_i^{\mathcal{P}}$
8: **if** $\mathcal{K}_{q_i}^{\mathcal{P}}$ is consistent **then**
9: **return** false
10: **else**
11: **return** true
12: **end if**

Compared to the deterministic version of the algorithm (Algorithm 4.3.1), we do not make the UNA but guess a partition of the individual names. We also non-deterministically choose one of the conjuncts that result from the transformation into CNF. For this conjunct, we guess an extended ABox and check whether the extended knowledge base for the guessed ABox is consistent and, therefore, a

counter-model for the query entailment.

In its (equivalent) negated form, Lemma 4.17 says that $\mathcal{K} \not\models q$ iff there is an extended knowledge base $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and q such that $\mathcal{K}_q$ is consistent. Together with Lemma 4.25 it follows, therefore, that Algorithm 4.3.2 is correct.

Data Complexity of Query Entailment in $\mathcal{SHIQ}$

We now analyse the data complexity of Algorithm 4.3.2 and show that deciding UCQ entailment in $\mathcal{SHIQ}$ is indeed in co-NP for data complexity. We assume that all concepts that occur in concept conjuncts of the input query or ABox assertions are literals. This is without loss of generality as we have shown in Section 2.4.

Theorem 4.28. *Let $\mathcal{T}$ be a $\mathcal{SHIQ}$ TBox, $\mathcal{R}$ a $\mathcal{SHIQ}$ role hierarchy, and q a union of Boolean conjunctive queries, and $\mathcal{A}$ a $\mathcal{SHIQ}$ ABox with $m_a = |\mathcal{A}|$. Algorithm 4.3.2 decides in non-deterministic polynomial time in m_a whether $\mathcal{K} \not\models q$ for $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$.*

Proof. Clearly, the size of an ABox $\mathcal{A}^{\mathcal{P}}$ in an $\mathcal{A}$-partition is bounded by m_a. As established in Lemma 4.26, the maximal size of an extended ABox $\mathcal{A}_q^{\mathcal{P}}$ is polynomial in m_a. Hence, $|\mathcal{A}^{\mathcal{P}} \cup \mathcal{A}_q^{\mathcal{P}}| \leq p(m_a)$ for some polynomial p. Due to Lemma 4.15 and since the size of q, $\mathcal{T}$, and $\mathcal{R}$ are assumed to be fixed, the sets $\mathsf{tree}_{\mathcal{K}^{\mathcal{P}}}(q_i)$ and $\mathsf{ground}_{\mathcal{K}^{\mathcal{P}}}(q_i)$ for each i such that $1 \leq i \leq \ell$ can be computed in time polynomial in m_a. From Lemma 4.23, we know that the translation of an extended knowledge base into an $\mathcal{ALCQI}b$ knowledge base is polynomial in m_a and the runtime of the algorithm for deciding consistency of an $\mathcal{ALCQI}b$ knowledge base [132] is also polynomial in m_a. □

The bound given in Theorem 4.28 is tight since the data complexity of conjunctive query entailment is already co-NP-hard for the $\mathcal{ALE}$ fragment of $\mathcal{SHIQ}$ [114].

Corollary 4.29. *Conjunctive query entailment in $\mathcal{SHIQ}$ is data complete for co-*NP*.*

4.3.3 Consequential Results

Due to the correspondence between query containment, query answering, and query entailment [22], the algorithm can also be used to decide containment of

two unions of conjunctive queries over a $\mathcal{SHIQ}$ knowledge base, which gives the following result:

Corollary 4.30. *Given a $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ and two unions of conjunctive queries q and q', the problem whether $\mathcal{K} \models q \subseteq q'$ is decidable.*

By using [106, Thm. 11], we further show that the consistency of a $\mathcal{SHIQ}$ knowledge base extended with (weakly-safe) Datalog rules (cf. Section 3.5) is decidable.

Corollary 4.31. *The consistency of $\mathcal{SHIQ}$+log-KBs (both under FOL semantics and under NM semantics) is decidable.*

4.4 Summary

With the decision procedure presented for entailment of unions of conjunctive queries in $\mathcal{SHIQ}$, we close a long standing open problem. The solution has immediate consequences on related areas, as it shows that several other open problems such as query answering, query containment and the extension of a KB with weakly safe Datalog rules for $\mathcal{SHIQ}$ are decidable as well. Regarding combined complexity, we present a deterministic algorithm that needs time single exponential in the size of the knowledge base and double exponential in the size of the query, which gives a tight upper bound for the problem. This result shows that deciding conjunctive query entailment is strictly harder than instance checking for $\mathcal{SHIQ}$. We further prove co-NP-completeness for data complexity. Interestingly, this shows that, regarding data complexity, deciding UCQ entailment is (at least theoretically) not harder than instance checking for $\mathcal{SHIQ}$, which was also a previously open question.

Chapter 5

Query Entailment for $\mathcal{SHOQ}$

So far, none of the conjunctive query answering techniques [22, 73, 86, 97, 129] is able to handle nominals. In this chapter, we address this issue and present a decision procedure for entailment of unions of conjunctive queries in $\mathcal{SHOQ}$.

The Description Logic $\mathcal{SHOQ}$ is obtained from $\mathcal{SHOIQ}$ by disallowing inverse roles. More precisely, for $\mathcal{S} = (N_C, N_R, N_I)$ a signature, the set of $\mathcal{SHOQ}$-*concepts* is the smallest set built inductively over $\mathcal{S}$ using the following grammar, where $o \in N_I, A \in N_C, n \in \mathbb{N}_0$, $r \in N_R$ is a role name and $s \in N_R$ is the name of a simple role:

$$C ::= \top \mid \bot \mid A \mid \{o\} \mid \neg A \mid \neg\{o\} \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall r.C \mid \exists r.C \mid \leqslant n\ s.C \mid \geqslant n\ s.C.$$

Since, in the presence of nominals, the ABox can be internalised [115], we assume that a $\mathcal{SHOQ}$ knowledge base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{R})$ over a signature $\mathcal{S} = (N_C, N_R, N_I)$, where $\mathcal{T}$ is a TBox and $\mathcal{R}$ is a role hierarchy. We use $\mathsf{nom}(\mathcal{K})$ for the set of individual names that occur in $\mathcal{K}$ and we assume that $\mathsf{nom}(\mathcal{K})$ is non-empty without further notice. This is without loss of generality since, otherwise, we can always add an axiom $\{o\} \sqsubseteq \top$ to the TBox for a fresh nominal $o \in N_I$.

When we use the term conjunctive query in this section, we implicitly mean $\mathcal{SHOQ}$ conjunctive query. Hence, we allow for $\mathcal{SHOQ}$-concepts (and, thus, nominals) in concept conjuncts and only role names in role conjuncts. Inverse roles are, however, implicitly still possible since if we want to query for all y such that y has an r^--successor x, which in $\mathcal{SHIQ}$ can be done by using the role conjunct $\mathsf{Inv}(r)(y, x)$, we can use the role conjunct $r(x, y)$ in which the position of the variables is swapped.

In order to simplify the notation, we do not allow for constants (individual names) in the query. In the presence of nominals this is clearly without loss of generality since, for each constant $o \in N_I$ that occurs as a term in a query q, we can replace all occurrences of o with a fresh variable $x_o \in N_V$ and add an additional concept conjunct $(\{o\})(x_o)$ to q.

The presented algorithm uses similar ideas to the decision procedure for $\mathcal{SHIQ}$ from the previous chapter, but the rewriting steps are slightly different due to the different structure of canonical models in $\mathcal{SHOQ}$. We try, however, to make the chapter self-contained by giving full definitions and proofs instead of just pointing out the difference to the case of $\mathcal{SHIQ}$. We add references to the previous chapter, where it is relevant. In the case of $\mathcal{SHOQ}$, the rewritten queries consist only of concept conjuncts, and we again use these concepts to reduce the task of deciding query entailment to the task of testing the consistency of extended $\mathcal{SHOQ}^{\sqcap}$ knowledge bases.

Several expressive Description Logics have been extended with role conjunction [71, 72, 118, 142] and implementations of such logics are readily available. The available extensions are, however, for logics that are more expressive than we require (they support logics with full role negation and Boolean role operators, which do not have the forest model property). This makes it difficult to obtain tight upper complexity bounds from them. Also the existing decision procedures for $\mathcal{SHOQ}$ [61] or $\mathcal{SHOIQ}$ [63, 78] are unnecessarily complicated for our purpose and are not worst-case optimal. Although knowledge base consistency checking for $\mathcal{SHOIQ}$ is known to be a NExpTime-complete problem [132], the algorithms run in nondeterministic double exponential time [63] and deterministic triple exponential time [78]. It is, therefore, unlikely that a worst-case optimal decision procedure for $\mathcal{SHOQ}^{\sqcap}$ can be obtained from these algorithms without major modifications. The decision procedure for $\mathcal{SHOQ}$ [61] is tableau based and, as most tableau based decision procedures, not worst-case optimal due to the introduction of nondeterminism. It was always conjectured that knowledge base consistency for $\mathcal{SHOQ}$ can be decided in ExpTime, as it is the case for $\mathcal{SHIQ}$, and we show that this is indeed the case by devising a novel automata based decision procedure that runs in deterministic exponential time for $\mathcal{SHOQ}$ input knowledge bases and in 2ExpTime for $\mathcal{SHOQ}^{\sqcap}$ knowledge bases.

5.1 Forest Bases and Canonical Interpretations

As for $\mathcal{SHIQ}$, our first goal is to show that we can restrict our attention to the canonical models of a knowledge base. Since the logic now allows for nominals, the forest structure of the canonical models is no longer as obvious. We can, however, still use a domain that builds a collection of trees. In the underlying forest bases, the elements within a tree can be related to their direct successors (in $\mathcal{SHIQ}$ we use neighbour due to the presence of inverse roles) and to some root/nominal nodes. The latter is not allowed for $\mathcal{SHIQ}$ since such relations represent links back to nominals that are enforced by concepts such as $\exists r.\{o\}$. Hence, the definition of canonical models and forest bases for $\mathcal{SHOQ}$ is similar to the one given in Chapter 4 for $\mathcal{SHIQ}$, but takes this additional case into account. In order to simplify the following definition, we again make the unique name assumption first and show later that this is without loss of generality.

Definition 5.1. Given a set of elements $R = \{r_1, \ldots, r_n\}$, a *forest F w.r.t. R* is a subset of $R \times \mathbb{N}^*$ such that, for each $r_i \in R, (r_i, \varepsilon) \in F$ and the set $\{w \mid (r_i, w) \in F\}$ is a tree.

Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base. A *forest base for $\mathcal{K}$* is an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ that interprets transitive roles in an unrestricted (i.e., not necessarily transitive) way and, additionally, satisfies the following conditions:

T1 $\Delta^{\mathcal{J}}$ is a forest w.r.t. $\mathsf{nom}(\mathcal{K})$;

T2 if $((o, w), (o', w')) \in r^{\mathcal{J}}$, then either

(a) $w' = \varepsilon$ or

(b) $o = o'$ and w' is a successor of w;

T3 for each $o \in \mathsf{nom}(\mathcal{K})$, $o^{\mathcal{J}} = (o, \varepsilon)$.

An interpretation $\mathcal{I}$ is *canonical for $\mathcal{K}$* if there exists a forest base $\mathcal{J}$ for $\mathcal{K}$ such that $\mathcal{I}$ is identical to $\mathcal{J}$ except that, for all non-simple roles r, we have

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq_{\mathcal{R}} r,\, s \in N_{tR}} (s^{\mathcal{J}})^+.$$

In this case, we say that $\mathcal{J}$ is a forest base *for* $\mathcal{I}$ and, if $\mathcal{I} \models \mathcal{K}$, we say that $\mathcal{I}$ is a *canonical model for $\mathcal{K}$*. △

We use the following Boolean query and knowledge base as a running example in this chapter:

Example 5.2. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}$ knowledge base with $t, t' \in N_{tR}$

$$\begin{aligned}\mathcal{T} = \{\ & \{o\} \sqsubseteq \exists t.(C \sqcap \exists r.(\exists r.(D \sqcap \exists t.(\{o\})))) \\ & \{o'\} \sqsubseteq \exists s.\top \sqcap \exists s.(\{o\})\} \\ \mathcal{R} = \{\ & r \sqsubseteq t'\}\end{aligned}$$

and $q = \{C(x), D(z), t'(x, z), t(z, x), r(x, y), r(y, z)\}$ with $\mathsf{Vars}(q) = \{x, y, z\}$.

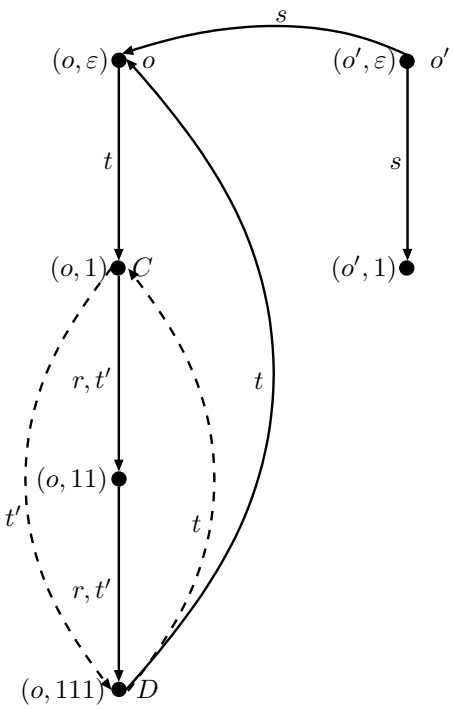

Figure 5.1: A representation of a canonical interpretation $\mathcal{I}$ for $\mathcal{K}$. The transitive shortcuts are shown as dashed lines; without them, the figure would show a representation of a forest base for $\mathcal{I}$.

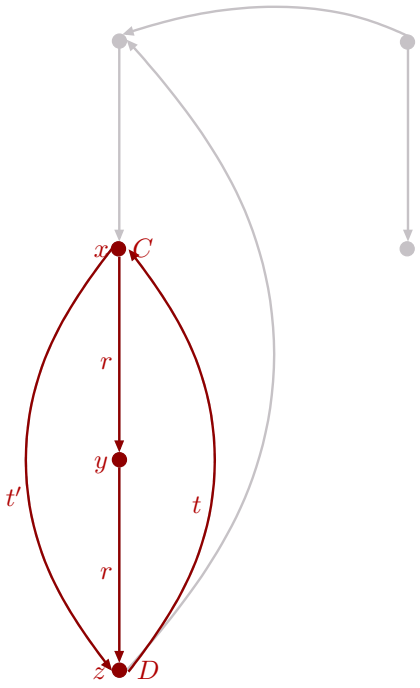

Figure 5.2: A graphical representation of the query q with a match in the canonical model (shown in grey, and without labels).

Figure 5.1 shows a graphical representation of a canonical model for $\mathcal{K}$ and Figure 5.2 shows a representation of the query used in the running example. It is not hard to check that $\mathcal{I} \models^{\pi} q$ for $\pi\colon \{x \mapsto (o, 1), y \mapsto (o, 11), z \mapsto (o, 111)\}$ and that also $\mathcal{K} \models q$.

The following lemma justifies our focus on canonical models.

Lemma 5.3. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base and q a union of connected Boolean conjunctive queries. $\mathcal{K} \not\models q$ iff there is some canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

Proof. The "if" direction is trivial.

The proof of the "only if" direction is very similar to the one for $\mathcal{SHIQ}$ (cf. Lemma 4.2). In order to make this chapter still relatively self contained, we present the adapted construction of a canonical model here and refer to the previous chapter for a detailed proof of the fact that the construction indeed yields a model for $\mathcal{K}$. Where it is relevant, we point out differences in the construction.

Since an inconsistent knowledge base entails every query, we can assume that $\mathcal{K}$ is consistent. Hence, there is an interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ such that $\mathcal{I}' \models \mathcal{K}$ and $\mathcal{I}' \not\models q$. From $\mathcal{I}'$, we construct a canonical model $\mathcal{I}$ for $\mathcal{K}$ and its forest base $\mathcal{J}$ as follows: we define the set $P \subseteq (\Delta^{\mathcal{I}'})^*$ of *paths* to be the smallest set such that

- for all $o \in \mathsf{nom}(\mathcal{K}), o^{\mathcal{I}'}$ is a path;
- $d_1 \cdots d_n \cdot d$ is a path, if
  - $d_1 \cdots d_n$ is a path,
  - $(d_n, d) \in r^{\mathcal{I}'}$ for some role name r,
  - if there is no $o \in \mathsf{nom}(\mathcal{K})$ such that $d = o^{\mathcal{I}'}$.

Now fix a set $S \subseteq \mathsf{nom}(\mathcal{K}) \times \mathbb{N}^*$ and a bijection $f \colon S \to P$ such that, for each $o \in \mathsf{nom}(\mathcal{K})$,

(i) $(o, \varepsilon) \in S$,

(ii) $\{w \mid (o, w) \in S\}$ is a tree,

(iii) $f(o, \varepsilon) = o^{\mathcal{I}'}$,

(iv) if $(o, w), (o, w') \in S$ with w' a successor of w, then $f(o, w') = f(o, w) \cdot d$ for some $d \in \Delta^{\mathcal{I}'}$.

For all $(o, w) \in S$, set $\mathsf{Tail}(o, w) = d_n$ if $f(o, w) = d_1 \cdots d_n$. Now, define a forest base $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ for $\mathcal{K}$ as follows:

(a) $\Delta^{\mathcal{J}} = S$;

(b) for each $o \in \mathsf{nom}(\mathcal{K})$, $o^{\mathcal{J}} = (o, \varepsilon) \in S$;

(c) for each $A \in N_C$ and $(o, w) \in S$, $(o, w) \in A^{\mathcal{J}}$ iff $\mathsf{Tail}(o, w) \in A^{\mathcal{I}'}$;

(d) for each $r \in N_R$, $((o, w), (o', w')) \in r^{\mathcal{J}}$ iff either

(I) $w' = \varepsilon$ and $(\mathsf{Tail}(o, w), o'^{\mathcal{I}'}) \in r^{\mathcal{I}'}$ or

(II) $o = o', w'$ is a successor of w, and $(\mathsf{Tail}(o, w), \mathsf{Tail}(o', w')) \in r^{\mathcal{I}'}$.

Please note that, compared to the construction for $\mathcal{SHIQ}$, Condition (d)(I) only requires that $w' = \varepsilon$, i.e., we can have relations from elements within a tree to any root node. Condition (d)(II) is now stricter since without inverse roles, we have to use successors and not neighbours as it is the case for $\mathcal{SHIQ}$.

The proof that $\mathcal{J}$ is indeed a forest base for $\mathcal{K}$ is quite long and, due to the close similarity of the argumentation with the proof of Lemma 4.2, we just refer back to the previous chapter here.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that is identical to $\mathcal{J}$ except that, for all non-simple roles r, we set

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq_{\mathcal{R}} r,\, s \in N_{tR}} (s^{\mathcal{J}})^{+}$$

Since $\mathcal{I} \models \mathcal{K}$ and $\mathcal{J}$ is a forest base for $\mathcal{I}$, $\mathcal{I}$ is a canonical model for $\mathcal{K}$.

Therefore, we only have to show that $\mathcal{I} \not\models q$. Let $q = q_1 \vee \ldots \vee q_n$ and assume to the contrary of what is to be shown that $\mathcal{I} \models q$. Then there is some π and i with $1 \leq i \leq n$ such that $\mathcal{I} \models^{\pi} q_i$. We now define a mapping $\pi' \colon \mathsf{Vars}(q_i) \to \Delta^{\mathcal{I}'}$ by setting $\pi'(x) = \mathsf{Tail}(\pi(x))$ for all $x \in \mathsf{Vars}(q_i)$. Since $\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of all concepts and $\mathcal{I}$ contains even less cycles than $\mathcal{I}'$, it is not hard to see that $\mathcal{I}' \models^{\pi'} q_i$ and, hence, $\mathcal{I}' \models^{\pi'} q$, which is a contradiction. □

5.2 Query Rewriting

For deciding whether a given UCQ is entailed by a $\mathcal{SHOQ}$ knowledge base, we transform each disjunct of the query in a four stage process into a set of $\mathcal{SHOQ}^{\sqcap}$ concepts, i.e., $\mathcal{SHOQ}$ with role conjunctions. We can then reduce the task of

deciding CQ entailment to the task of deciding $\mathcal{SHOQ}^{\sqcap}$ knowledge base consistency. As for $\mathcal{SHIQ}$, we show that if the query is entailed by the knowledge base, then, for any canonical model, there is a rewritten query that is forest-shaped and has a match in the canonical model. We will briefly introduce the rewriting steps by means of our running example before we give a formal definition.

The first step, *collapsing*, in which we can identify variables, is as for $\mathcal{SHIQ}$. The next two steps are similar to the split, loop, and forest rewriting steps, but we adapt the definitions more precisely to the situations that can occur in $\mathcal{SHOQ}$ canonical models. The second step is called *nominal rewriting* and, in a nominal rewriting, we can replace role conjuncts of the form $r(x, x')$ for which r is non-simple with two role conjuncts by possibly introducing a fresh variable. This allows for explicating all shortcuts that bypass a nominal. In our running example, we can choose to replace the conjunct $t(z, x)$, which bypasses the nominal node (o, ε) for the given mapping π and canonical model $\mathcal{I}$, with $t(z, x_r), t(x_r, x)$ for $x_r \in N_V$ a fresh variable. We call this nominal rewriting q_{nr}.

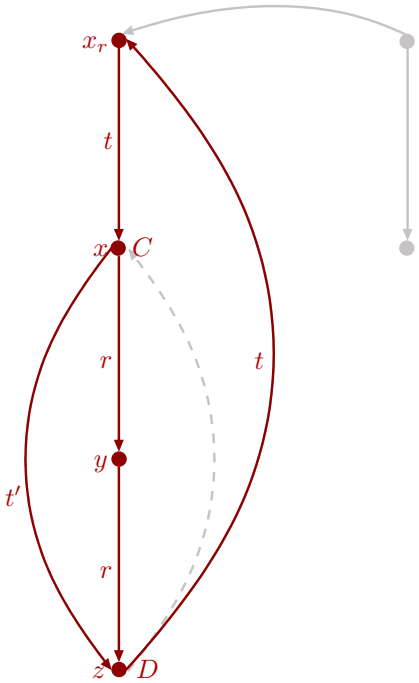

Figure 5.3: A graphical representation of the nominal rewriting q_{nr} for q. The canonical model $\mathcal{I}$ is shown in grey (without labels).

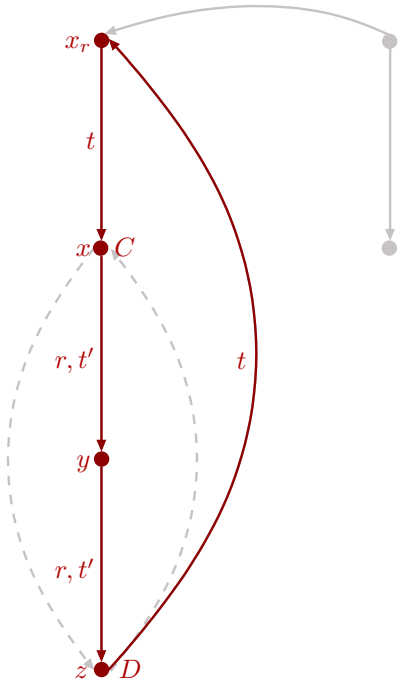

Figure 5.4: A graphical representation of the shortcut rewriting q_{sr} for q_{nr}. The canonical model $\mathcal{I}$ is shown in grey (without labels).

Figure 5.3 shows a graphical representation of q_{nr} and it is not hard to check that $\mathcal{I} \models^{\pi_{nr}} q_{nr}$, where π_{nr} is the extension of π that maps x_r to (o, ε). At the end of the nominal rewriting step, we also "guess", for each of the rewritten queries, which variables correspond to nominals.

In the third step, called *shortcut rewriting*, we explicate shortcuts within a tree such as $t'(x, z)$ in our running example, by replacing role conjuncts with a non-simple role with up to $\sharp(q)$ role conjuncts that use a transitive sub-role. In our running example, we can replace $t'(x, z)$ with $t'(x, y), t'(y, z)$, which yields a query that no longer uses any shortcuts in $\mathcal{I}$ with the given mapping (see Figure 5.4).

In the forth and last step, we filter out those queries that can still not be expressed as a concept. Those queries are trivially false since their structure cannot be mapped to the canonical models of the knowledge base. The remaining queries are transformed into concepts by applying the rolling-up technique.

As for $\mathcal{SHIQ}$, we may introduce concepts that contain an intersection of roles during the rolling-up. We define, therefore, the extension of $\mathcal{SHOQ}$ with role conjunction/intersection, denoted as $\mathcal{SHOQ}^{\sqcap}$ and, in the following section, we show how to decide the consistency of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases.

In addition to the constructors introduced for $\mathcal{SHOQ}$, $\mathcal{SHOQ}^{\sqcap}$ allows for concepts of the form

$$C ::= \forall R.C \mid \exists R.C \mid \leqslant n\ S.C \mid \geqslant n\ S.C,$$

where $R := r_1 \sqcap \ldots \sqcap r_n$, $S := s_1 \sqcap \ldots \sqcap s_n$, $r_1, \ldots, r_n$ are roles, and $s_1, \ldots, s_n$ are simple roles. The interpretation function is extended such that $(r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}} = {r_1}^{\mathcal{I}} \cap \ldots \cap {r_n}^{\mathcal{I}}$.

For our running example, we can guess, for example, that x_r corresponds to the nominal o. The query can then be expressed by the following concept:

$$\{o\} \sqcap \exists t.(C \sqcap \exists(r \sqcap t').(\exists(r \sqcap t').(D \sqcap \exists t.\{o\})))$$

Finally, we use the concepts from all possible rewritings and reduce the task of deciding query entailment to the task of deciding knowledge base consistency in a similar way as for $\mathcal{SHIQ}$ by augmenting $\mathcal{K}$ with an axiom of the form $\top \sqsubseteq \neg C_q$ for each obtained concept C_q.

Please note that in the following definitions, we assume that q is a conjunctive

query and not a union of conjunctive queries since we apply the rewriting steps to each disjunct separately.

Definition 5.4. Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base and q a Boolean conjunctive query. A *collapsing* q_{co} of q is obtained by adding zero or more equality conjuncts of the form $x \approx x'$ for $x, x' \in \mathsf{Vars}(q)$ to q. We use $\mathsf{co}(q)$ to denote the set of all queries that are a collapsing of q.

A *nominal rewriting* q_{nr} of q and $\mathcal{K}$ is obtained by choosing, for each role conjunct $r(x, y) \mathrel{\bar{\in}} q$ such that there is a role $s \in \mathsf{Trans}_{\mathcal{R}}$ and $s \sqsubseteq^*_{\mathcal{R}} r$ to either do nothing or to replace $r(x, y)$ with $s(x, z)$ and $s(z, y)$ for $z \in N_V$ a possibly fresh variable. We use $\mathsf{nr}_{\mathcal{K}}(q)$ to denote the set of pairs (q_{nr}, R) such that q_{nr} is a nominal rewriting of a query $q_{co} \in \mathsf{co}(q)$ and R is a subset of $\mathsf{Vars}(q_{nr})$. We call R a *root choice* for q_{nr}.

A *shortcut rewriting* of q and $\mathcal{K}$ is obtained from q by replacing each role conjunct $t(x_1, x_n)$ from q for which there is a sequence $r_1(x_1, x_2), \ldots, r_{n-1}(x_{n-1}, x_n) \mathrel{\bar{\in}} q$ and a role $s \in N_{tR}$ such that $s \sqsubseteq^*_{\mathcal{R}} t$ with $n-1$ role conjuncts $s(x_1, x_2), \ldots, s(x_{n-1}, x_n)$. We use $\mathsf{sr}_{\mathcal{K}}(q)$ to denote the set of all pairs (q_{sr}, R) such that there is a pair $(q_{nr}, R) \in \mathsf{nr}_{\mathcal{K}}(q)$ and q_{sr} is a shortcut rewriting of q_{nr}. △

We assume that $\mathsf{nr}_{\mathcal{K}}(q)$ contains no isomorphic queries, i.e., differences in (newly introduced) variable names only are neglected.

Readers might wonder why we do not delete the role conjunct $t(x_1, x_n)$ in a shortcut rewriting instead of replacing it $n - 1$ role conjuncts and require that $r_i \sqsubseteq^*_{\mathcal{R}} t$ for $1 \leq i < n$. For $\mathcal{SHOQ}$ this would be possible, and certainly advisable in practical implementations; we comment on this later in Remark 5.8. Roughly speaking, this would require a stricter definition of canonical models, which would make both the proofs and the definitions more complicated. Since our goal is mainly to show decidability, we only sketch what might be done in order to optimise the algorithm.

5.2.1 Query Shapes and Matches

We now show how we can filter out those queries that are trivially false since they have a structure that cannot occur in canonical models. For this, we define forest-shaped queries, similar to our forest-shaped canonical models. Intuitively, the variables in the root choice of a rewritten query correspond to nominals and a forest-shaped query can be mapped to a set of trees such that each variable in

the root choice is mapped to the root of a tree. All other variables are mapped such that, for each role conjunct $r(x, y)$ in the query, either the image of y is a successor of the image of x in one tree or y corresponds to a nominal and $r(x, y)$ corresponds to an edge back to some nominal.

Definition 5.5. A *tree mapping w.r.t. q* is a total function f from $\mathsf{Vars}(q)$ to a tree such that

1. f is bijective modulo $\overset{*}{\approx}$,
2. $r(x, y) \bar{\in} q$ implies that $f(y)$ is a successor of $f(x)$.

A query q is *tree-shaped* if there exists a tree mapping w.r.t. q.

A *forest mapping w.r.t. q and a root choice R* is a total function f from $\mathsf{Vars}(q)$ to a forest F w.r.t. R such that

1. f is bijective modulo $\overset{*}{\approx}$,
2. if $r(x, y) \in q$, then either
   (a) $y \in R$ or
   (b) there is some $(x_r, w) \in F$ and $c \in \mathbb{N}$ such that $x_r \in R$, $f(x) = (x_r, w)$, and $f(y) = (x_r, w \cdot c)$.

We say that q is *forest-shaped w.r.t. R* if either $R = \emptyset$ and q is tree-shaped or $R \neq \emptyset$ and there exists a forest mapping w.r.t. q and R.

We use $\mathsf{fr}_{\mathcal{K}}(q)$ to denote the set of all pairs $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ for which q_{sr} is forest-shaped w.r.t. R. △

Similarly to tree- and forest-shaped matches for $\mathcal{SHIQ}$ conjunctive queries, we also define tree- and forest-shaped matches on canonical models for $\mathcal{SHOQ}$ conjunctive queries.

Definition 5.6. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a canonical model for $\mathcal{K}$ such that $\mathcal{I} \models^{\pi} q$. The match π *induces* the root choice $R = \{x \mid \pi(x) = (o, \varepsilon) \text{ for some } o \in \mathsf{nom}(\mathcal{K})\}$. We call π a *tree match* if the induced root choice $R = \emptyset$ and there exists a total function f from $\mathsf{ran}(\pi)$ to a tree T such that

1. f is bijective modulo $\overset{*}{\approx}$ and
2. $r(x, y) \bar{\in} q$ implies that $f(\pi(y))$ is a successor of $f(\pi(x))$.

We call π a *forest match* if either π is a tree match or there is a total mapping f from $\mathsf{ran}(\pi)$ to a forest F w.r.t. the induced root choice R such that

1. f is bijective modulo $\overset{*}{\approx}$,
2. if $r(x, y) \in q$, then either
   (a) $y \in R$ or
   (b) there is some $(x_r, w) \in F$ and $c \in \mathbb{N}$ such that $f(\pi(x)) = (x_r, w)$ and $f(\pi(y)) = (x_r, w \cdot c)$.

△

The following lemma shows that we can indeed omit queries that are not forest-shaped w.r.t. their root choice.

Lemma 5.7. *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model for $\mathcal{K}$.*

(1) If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and a forest match π_{fr} such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ and R is the root choice induced by π_{fr}.

(2) If $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models q_{fr}$, then $\mathcal{I} \models q$.

Proof. For (1): Let π be such that $\mathcal{I} \models^{\pi} q$. First, we build a collapsing q_{co} of q by adding an conjunct $x \approx y$ for all variables $x, y \in \mathsf{Vars}(q)$ for which $\pi(x) = \pi(y)$. It is not hard to verify that q_{co} is indeed a collapsing of q, $\mathcal{I} \models^{\pi} q_{co}$, and that π is an injection modulo $\overset{*}{\approx}$.

We now identify the role conjuncts to which we need to apply the nominal rewriting step. For this, we first define the set of role conjuncts that conform with the conditions for forest matches. The nominal rewriting is then applied to the remaining role conjuncts: let $S \subseteq q_{co}$ be the set of all role conjuncts $r(x, y)$ such that, for $\pi(x) = (o, w)$ and $\pi(y) = (o', w')$, either

1. $w' = \varepsilon$ or
2. $o = o'$ and w is a proper prefix of w'.

Let $S_{nr} \subseteq q_{co}$ be the set of role conjuncts $r(x, y)$ such that $r(x, y) \notin S$. We now build a nominal rewriting of q_{co} by replacing each role conjunct $r(x, y) \in S_{nr}$ with two role conjuncts that contain a possibly new variable. This variable

corresponds, intuitively, to one of the roots. Since $\mathcal{SHOQ}$ does not allow for inverse roles, such relationships can only exist due to transitive roles and links back to a nominal node.

More precisely, let $\mathcal{J}$ be a forest base for $\mathcal{I}$. We show that, for each $r(x,y) \in S_{nr}$ with $\pi(x) = (o,w)$ and $\pi(y) = (o',w')$, there exists a role $s \in N_{tR}$ such that $s \sqsubseteq^*_{\mathcal{R}} r, (\pi(x),(o',\varepsilon)) \in s^{\mathcal{I}}$, and $((o',\varepsilon),\pi(y)) \in s^{\mathcal{I}}$. Since $\mathcal{I} \models^{\pi} q_{co}$, we have $(\pi(x),\pi(y)) \in r^{\mathcal{I}}$. Since $r(x,y) \notin S$, either

1. $o \neq o'$ and $w \neq \varepsilon$ or
2. $o = o'$ and w' is not a proper prefix of w.

Because $\mathcal{J}$ is a forest base, this implies that $(\pi(x),\pi(y)) \notin r^{\mathcal{J}}$. It follows that there is a sequence $d_1,\dots,d_n \in \Delta^{\mathcal{I}}$ and a role $s \in N_{tR}$ such that $d_1 = \pi(x), d_n = \pi(y), s \sqsubseteq^*_{\mathcal{R}} r$, and $(d_i,d_{i+1}) \in s^{\mathcal{J}}$ for $1 \leq i < n$. Moreover, by definition of forest bases, there is an i with $1 \leq i \leq n$ such that $d_i = (o',\varepsilon)$.

For each $r(x,y) \in S_{nr}$, we select an appropriate element d_i and a role s as described above, denote the former with $d_{o'}$, the latter with s_{xRy}, and obtain the query q_{nr} from q_{co} by doing the following:

- if $d_{o'} = \pi(z)$ for some $z \in \mathsf{Vars}(q_{co})$, then replace $r(x,y)$ with $s_{xRy}(x,z)$ and $s_{xRy}(z,y)$;
- otherwise, introduce a new variable $v_{o'} \in N_V$ and replace $r(x,y)$ with $s_{xRy}(x,v_{o'})$ and $s_{xRy}(v_{o'},y)$.

It is not hard to check that q_{nr} is indeed a nominal rewriting of q_{co}.

Let π_{fr} be the extension of π that maps each newly introduced variable $v_{o'}$ to $d_{o'}$. It is easily seen that q_{nr} is connected, π_{fr} is injective modulo $\overset{*}{\approx}$, and $\mathcal{I} \models^{\pi_{fr}} q_{nr}$. The match π_{fr} induces the root choice

$$R = \{x \mid \pi_{fr}(x) = (o,\varepsilon) \text{ for some } o \in \mathsf{nom}(\mathcal{K})\}.$$

Since q_{nr} is a nominal rewriting of q_{co} and by definition of the set $\mathsf{nr}_{\mathcal{K}}$, the pair (q_{nr},R) is in $\mathsf{nr}_{\mathcal{K}}(q)$. If π_{fr} is already a forest match w.r.t. R, we are done since a query for which there is a forest match is forest-shaped w.r.t. R. Assume, therefore, that π_{fr} is not a forest match. Since we already dealt with all conjuncts that are mapped from one tree into another one or that used a relationship to an ancestor within the same tree, the only remaining situation in which π_{fr}

is not a forest match is that we have a shortcut to a descendant and, hence, an undirected cycle. More precisely, there are conjuncts $r_1(x_1, x_2), \ldots, r_{n-1}(x_{n-1}, x_n)$ and $t(x_1, x_n)$ in q_{nr} with $\pi_{fr}(x_i) = (o, w_i)$ and w_i is a proper prefix of w_{i+1} for each i with $1 \leq i < n$. We show that $n > 2$ implies that there is a role s such that $s \in N_{tR}, s \sqsubseteq^*_{\mathcal{R}} t$ and $(\pi_{fr}(x_i), \pi_{fr}(x_{i+1})) \in s^{\mathcal{I}}$ for each i with $1 \leq i < n$. Since $\mathcal{I} \models^{\pi_{fr}} q_{nr}$, we have $(\pi_{fr}(x_1), \pi_{fr}(x_n)) \in t^{\mathcal{I}}$ and $(\pi_{fr}(x_i), \pi_{fr}(x_{i+1})) \in r_i{}^{\mathcal{I}}$ for each i with $1 \leq i < n$. Since $\mathcal{J}$ is a forest base and w_i is a proper prefix of $w_{i+1}, n > 2$ implies that $(\pi_{fr}(x_1), \pi_{fr}(x_n)) \notin t^{\mathcal{J}}$. It follows that there is a unique shortest sequence $d_1, \ldots, d_m \in \Delta^{\mathcal{I}}$ and a role $s \in N_{tR}$ such that $d_1 = \pi_{fr}(x_1), d_m = \pi_{fr}(x_n), s \sqsubseteq^*_{\mathcal{R}} t$, and $(d_i, d_{i+1}) \in s^{\mathcal{J}}$ for $1 \leq i < m$. By definition of canonical models and since w_i is a proper prefix of w_{i+1} for each i with $1 \leq i < n$, each (o, w_i) occurs in the sequence $d_1, \ldots, d_m$ and, since $s \in N_{tR}$, $((o, w_i), (o, w_{i+1})) \in s^{\mathcal{I}}$ for each i with $1 \leq i < n$. For each $t(x_1, x_n)$ as described above, we select an appropriate role s and replace $t(x_1, x_n)$ with $n - 1$ role conjuncts $s(x_1, x_2), \ldots, s(x_{n-1}, x_n)$ and call the resulting query q_{sr}.

It is again not hard to check that q_{sr} is indeed a shortcut rewriting of q_{nr} and that, hence, $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$. Furthermore, π_{fr} is now a forest match w.r.t. R, which implies that q_{sr} is forest-shaped w.r.t. R. Hence, $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ for $q_{fr} = q_{sr}$ as required.

For (2): Since $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q), (q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ for $q_{sr} = q_{fr}$. Furthermore, there is a pair $(q_{nr}, R) \in \mathsf{nr}_{\mathcal{K}}(q)$ such that q_{sr} is a shortcut rewriting of q_{nr}. In a shortcut rewriting, a role conjunct $t(x_1, x_n)$ is replaced with a sequence of role conjuncts $s(x_1, x_2), \ldots, s(x_{n-1}, x_n)$ such that s is a transitive sub-role of t. Hence, $\mathcal{I} \models q_{sr}$ implies $\mathcal{I} \models q_{nr}$. Similarly, $(q_{nr}, R) \in \mathsf{nr}_{\mathcal{K}}(q)$ implies that there is a collapsing $q_{co} \in \mathsf{co}(q)$ such that q_{nr} is a nominal rewriting of q_{co}. Repeating the same argument as for the shortcut rewriting, we have that $\mathcal{I} \models q_{nr}$ implies $\mathcal{I} \models q_{co}$. Since q_{co} is a collapsing of q and since q is a subset of q_{co}, we trivially have that $\mathcal{I} \models q$ as required. □

Remark 5.8. As promised above, we comment on why we do not allow the complete deletion of role conjuncts in a shortcut rewriting by requiring, additionally, that $r_i \sqsubseteq^*_{\mathcal{R}} s$ for $1 \leq i < n$. For example, let $q = \{r_1(x_1, x_2), r_2(x_2, x_3), t(x_1, x_3)\}$ and $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ with $t \in N_{tR}, \mathcal{R} = \emptyset$, and

$$\mathcal{T} = \{\{o\} \sqsubseteq \exists r_1.(\exists r_2.\top) \sqcap \exists t.(\exists t.\top)\}.$$

With the stricter definition, the query has no shortcut rewriting and is not forest-shaped. Hence, we directly (and correctly) conclude $\mathcal{K} \not\models q$. Unfortunately, Lemma 5.7 does not hold with the stricter definition. For example, let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a canonical model for $\mathcal{K}$ as depicted in Figure 5.5. It is not hard to check that $\mathcal{I} \models q$ and that, hence, according to the above lemma there should be a forest-shaped shortcut rewriting of q, which is not the case. In order for the above lemma to hold, we need a stricter definition of canonical models. In the existing definition of canonical models (Definition 5.1), we neither minimise the interpretation of roles nor maximise the number of successors by splitting successors wherever possible. For example, let $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be an alternative canonical model for $\mathcal{K}$ as depicted in Figure 5.6. In fact, $\mathcal{I}'$ is exactly homomorphic to a model that the tableau algorithm for $\mathcal{SHOQ}$ [61] would generate since, for each existential quantifier or atleast number restriction, we introduce new successors and we merge successors only when necessary due to atmost number restrictions. In order to use the optimised definition of shortcut rewritings, we would need a similar requirement for our canonical models, which is, however, more complicated to formalise.

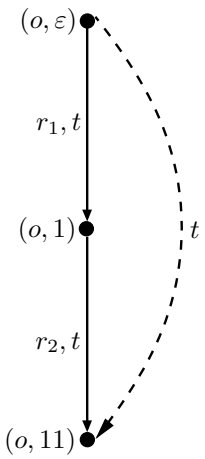

Figure 5.5: A representation of a canonical model $\mathcal{I}$ for $\mathcal{K}$.

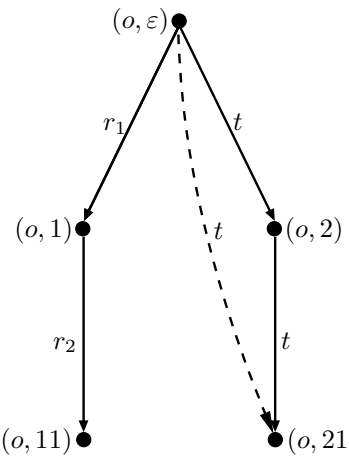

Figure 5.6: A representation of an alternative canonical model $\mathcal{I}'$ for $\mathcal{K}$.

5.2.2 From Forest-Shaped Queries to Concept Conjuncts

We now build a query that consists only of concept conjuncts for each $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ by replacing the variables from R with nominals from $\mathsf{nom}(\mathcal{K})$ and applying

the rolling-up technique.

Definition 5.9. Let $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$. A *grounding for q_{fr} w.r.t. R* is a total function $\tau\colon R \to \mathsf{nom}(\mathcal{K})$ that is injective modulo $\overset{*}{\approx}$. We build $\mathsf{con}(q_{fr}, R, \tau)$ as follows:

1. For each $r(x, x_r) \,\bar{\in}\, q_{fr}$ with $x_r \in R$, replace $r(x, x_r)$ with $(\exists r.\{\tau(x_r)\})(x)$.
2. For each $x_r \in R$, add a concept conjunct $(\{\tau(x_r)\})(x_r)$ to q_{fr}.
3. Call the result of 1 and 2 q.
4. We now inductively assign, to each $x \in \mathsf{Vars}(q)$ a concept $\mathsf{con}(x)$ as follows:
   - if there is no role conjunct $r(x, x') \,\bar{\in}\, q$, then $\mathsf{con}(x) = \bigsqcap_{C(x) \bar{\in} q} C$,
   - if there are role conjuncts $r(x, x_1), \ldots, r(x, x_k) \,\bar{\in}\, q$, then

$$\mathsf{con}(x) = \bigsqcap_{C(x) \bar{\in} q} C \sqcap \bigsqcap_{1 \leqslant i \leqslant k} \exists\big(\bigsqcap_{r(x,x_i) \bar{\in} q} r\big).\mathsf{con}(x_i).$$

5. Finally, $\mathsf{con}(q_{fr}, R, \tau) = \{(\mathsf{con}(x))(x) \mid x \in \mathsf{Vars}(q)$ and there is no role conjunct $r(x', x) \,\bar{\in}\, q\}$.

We use $\mathsf{con}_{\mathcal{K}}(q)$ for the set $\{\mathsf{con}(q_{fr}, R, \tau) \mid (q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and τ is a grounding w.r.t. $R\}$. △

Please note that, after the first step, the resulting query consists of a set of unconnected components such that each component is a tree-shaped query with a distinguished root variable. This root variable must not necessarily belong to the root choice R. In Step 4, we collect all query concepts for these root variables in the set $\mathsf{con}(q_{fr}, R, \tau)$. Hence $\mathsf{con}(q_{fr}, R, \tau)$ is a conjunctive query of the form $\{C_1(x_1), \ldots, C_n(x_n)\}$ with $x_i \neq x_j$ for $1 \leq i < j \leq n$ and each C_i is a $\mathcal{SHOQ}^{\sqcap}$-concept.

Lemma 5.10. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base, q a Boolean conjunctive query, and $\mathcal{I}$ a model of $\mathcal{K}$.*

(1) If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is some $\mathsf{con}(q_{fr}, R, \tau) \in \mathsf{con}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models \mathsf{con}(q_{fr}, R, \tau)$.

(2) If $\mathcal{I} \models \mathsf{con}(q_{fr}, R, \tau)$ for some $\mathsf{con}(q_{fr}, R, \tau) \in \mathsf{con}_{\mathcal{K}}(q)$, then $\mathcal{I} \models q$.

Proof. For (1): By assumption, $\mathcal{I}$ is canonical and $\mathcal{I} \models q$. By Lemma 5.7 there is a pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and a forest match π_{fr} such that R is the root choice induced by π_{fr} and $\mathcal{I} \models^{\pi_{fr}} q_{fr}$. Let τ be such that each $x \in R$ is mapped to $\pi_{fr}(x)$, where $\pi_{fr}(x) = (o, \varepsilon)$ for some $o \in \mathsf{nom}(\mathcal{K})$. Such a mapping τ exists since R is the root choice induced by π_{fr}. By definition of the semantics, we clearly have that $I \models q'$ for q' the query obtained after the first two steps of building $\mathsf{con}(q_{fr}, R, \tau)$. Since q' is a collection of (directed) trees, each with a unique root variable, Lemma 4.7 easily transfers from $\mathcal{SHIQ}^{\sqcap}$ to $\mathcal{SHOQ}^{\sqcap}$ and we get that $\mathcal{I} \models \mathsf{con}(q_{fr}, R, \tau)$.

For (2): We can again proceed in a similar way as in the proof of Lemma 4.7 to show that $\mathcal{I} \models \mathsf{con}(q_{fr}, R, \tau)$ implies $\mathcal{I} \models q_{fr}$. Together with Lemma 5.7 this proves the claim. □

5.2.3 Correctness of the Rewriting Steps

We now show that the union of the queries in $\mathsf{con}_{\mathcal{K}}(q)$ can be used to decide entailment of q and that there is a bound on the cardinality of this set. We can then use the standard methods for deciding entailment of tree-shaped conjunctive queries in order to decide entailment of arbitrary conjunctive queries in $\mathcal{SHOQ}$.

Theorem 5.11. *Let* $\{q_1, \ldots, q_\ell\} = \mathsf{con}_{\mathcal{K}}(q)$. *Then* $\mathcal{K} \models q$ *iff* $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$.

Proof. For the "if" direction: by assumption, $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$. Hence, for each model $\mathcal{I}$ of $\mathcal{K}$, there is a query q_i with $1 \leqslant i \leqslant \ell$ such that $\mathcal{I} \models q_i$. Since $q_i \in \mathsf{con}_{\mathcal{K}}(q)$, it is of the form $\mathsf{con}(q_{fr}, R, \tau)$ and, by Lemma 5.10 it follows that $\mathcal{I} \models q$.

For the "only if" direction: by Lemma 5.3 (in its negated form) we have that $\mathcal{K} \models q$ iff all canonical models $\mathcal{I}$ of $\mathcal{K}$ are such that $\mathcal{I} \models q$. Hence, we can restrict our attention to the canonical models of $\mathcal{K}$. Now let $\mathcal{I}$ be a canonical model of $\mathcal{K}$ and assume that $\mathcal{I} \models q$. Hence, by Lemma 5.10, there is some $q_i = \mathsf{con}(q_{fr}, R, \tau) \in \mathsf{con}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models q_i$ as required. □

We now give upper bounds on the size and number of queries in $\mathsf{con}_{\mathcal{K}}(q)$. Obviously, the number of conjuncts in a query is bounded by its size, hence $\sharp(q) \leq |q|$ and, for simplicity, we use n as the size and the cardinality of q in what follows.

Lemma 5.12. *Let* q *be a Boolean conjunctive query,* $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ *a* $\mathcal{SHOQ}$ *knowledge base,* $|q| = n$ *and* $|\mathcal{K}| = m$. *Then there is a polynomial* p *such that*

1. $\sharp(\mathsf{co}(q)) \leq 2^{p(n)}$ *and, for each* $q' \in \mathsf{co}(q)$, $|q'| \leq p(n)$,
2. $\sharp(\mathsf{nr}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{nr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,
3. $\sharp(\mathsf{sr}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{sr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,
4. $\sharp(\mathsf{con}_{\mathcal{K}}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{con}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$.

Proof.

1. Since the collapsing step is the same for $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$, the proof of Lemma 4.15 directly applies to this claim.

2. For each of the at most n role conjuncts, we can choose to do nothing or replace the conjunct with two conjuncts. For every replacement, we can choose to introduce a new variable or re-use one of the at most n existing variables. If we introduce a new variable every time, the new query contains at most $2n$ variables. Since $\mathcal{K}$ contains at most m non-simple roles that are a sub-role of a role used in role conjuncts of q, we have at most m roles to choose from when replacing a role conjunct. Overall, this gives us at most $1 + m(n + 1)$ choices for each of the at most n role conjuncts in a query and, therefore, the number of nominal rewritings for each query $q' \in \mathsf{co}(q)$ is polynomial in m and exponential in n. Since the number of variables in a nominal rewriting is linear in the number of variables in q, the number of root choices for each nominal rewriting is at most exponential in n and, in combination with the results from (1), this also shows that the cardinality of the set $\mathsf{nr}_{\mathcal{K}}(q)$ is polynomial in m and exponential in n.

   Since we add at most one new role conjunct for each of the existing role conjuncts, the size of a query $q' \in \mathsf{nr}_{\mathcal{K}}(q)$ is linear in n.

3. In a shortcut rewriting, each role conjunct $r(x, y)$ can be replaced with at most $n - 1$ role conjuncts that use one of the at most m transitive sub-roles of r. Hence, whenever this step is applicable to a role conjunct, we have m choices and, together with the bounds on the cardinality of $\mathsf{nr}_{\mathcal{K}}(q)$, this shows that the cardinality of the set $\mathsf{sr}_{\mathcal{K}}(q)$ is exponential in n and polynomial in m.

   Since the number of role conjuncts that we can introduce for each existing role conjunct is bounded by $n - 1$, the size of each query $q' \in \mathsf{sr}_{\mathcal{K}}(q)$ is at most quadratic in n.

4. By (1)-(3) above, the number of variables in a root choice is at most $2n$ and there are at most m nominal names occurring in $\mathcal{K}$ that can be used for the mapping τ from variables to nominal names. Hence the number of different ground mappings τ is at most polynomial in m and exponential in n. The number of ground queries that a single tuple $(q_{fr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ can contribute is, therefore, also at most polynomial in m and exponential in n. Together with the bound on the cardinality of $\mathsf{sr}_{\mathcal{K}}(q)$, this shows that the cardinality of $\mathsf{con}_{\mathcal{K}}(q)$ is polynomial in m and exponential in n. Again it is not hard to see that the size of each query $q' \in \mathsf{con}_{\mathcal{K}}(q)$ is polynomial in n.

□

As a consequence of the above lemma, there is a bound on the number of queries in $\mathsf{con}_{\mathcal{K}}(q)$ and it is not hard to see that this set can be computed in time polynomial in m and exponential in n.

Please note that each query $q' \in \mathsf{con}_{\mathcal{K}}(q)$ is a set of concept conjuncts of the form $\{C_1(x_1), \ldots, C_n(x_n)\}$, i.e., each q' contains n unconnected components. In the following, we assume for convenience that conjunctive queries are written as a conjunction of conjuncts and not in the set notation, e.g., we now write $C_1(x_1) \wedge \ldots \wedge C_n(x_n)$. By transforming the disjunction $q_1 \vee \ldots \vee q_\ell$ of queries in $\mathsf{con}_{\mathcal{K}}(q)$ into conjunctive normal form (cf. [129, 7.3.2]), we can reduce the problem of deciding whether $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$ to deciding whether $\mathcal{K}$ entails each union of connected conjunctive queries $\{co_1\} \vee \ldots \vee \{co_\ell\}$ such that co_i is a concept conjunct from q_i. Let $\mathsf{con}_{\mathcal{K}}(q) = \{q_1, \ldots, q_\ell\}$. We use $\mathsf{cnf}(\mathsf{con}_{\mathcal{K}}(q))$ for the conjunctive normal form of $q_1 \vee \ldots \vee q_\ell$. We now show how we can decide entailment of unions of conjunctive queries, where each conjunct consists of one concept conjunct only. This suffices to decide conjunctive query entailment for $\mathcal{SHOQ}$.

Definition 5.13. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}$ knowledge base, q a Boolean conjunctive query, and $C_1(x_1) \vee \ldots \vee C_\ell(x_\ell)$ a conjunct from $\mathsf{cnf}(\mathsf{con}_{\mathcal{K}}(q))$. An *extended knowledge base w.r.t. $\mathcal{K}$ and q* is a pair $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R})$ such that $\mathcal{T}_q = \{\top \sqsubseteq \neg C_i$ with $1 \leq i \leq \ell\}$. △

We can now use the extended knowledge bases in order to decide conjunctive query entailment:

Theorem 5.14. *$\mathcal{K} \models q$ iff each extended knowledge base $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and q is inconsistent.*

Please note that the extended knowledge bases are in $\mathcal{SHOQ}^{\sqcap}$. At this point, any reasoning algorithm for $\mathcal{SHOQ}^{\sqcap}$ can be used for deciding query entailment. In the following section, we present one such decision procedure and we analyse the complexity of the presented algorithm. As a first step, we analyse here the bounds on the number of extended knowledge bases and their size.

Lemma 5.15. *Let q be a Boolean conjunctive query with $|q| = n$ and $\mathcal{K}$ a $\mathcal{SHOQ}$ knowledge base with $|\mathcal{K}| = m$. There is a polynomial p such that the number of extended knowledge bases w.r.t. $\mathcal{K}$ and q is bounded by $2^{p'(m)\cdot 2^{p'(n)}}$, the size of each extended knowledge base is polynomial in m and exponential in n, and the longest role conjunction occurring in an extended knowledge base is polynomial in n.*

Proof. By Lemma 5.12 (4) there is a polynomial p such that the cardinality of $\mathsf{con}_{\mathcal{K}}(q)$ is bounded by $2^{p(n)\cdot \log p(m)}$. The size of each query in $\mathsf{con}_{\mathcal{K}}(q)$ and, hence, the number of its concept conjuncts is polynomial in n. Hence, there is a polynomial p' such that $\mathsf{cnf}(\mathsf{con}_{\mathcal{K}}(q))$ contains at most $2^{p'(m)\cdot 2^{p'(n)}}$ conjuncts and the number of disjuncts in each of the conjuncts is exponential in n and polynomial in m. This means that we have to test at most $2^{p'(m)\cdot 2^{p'(n)}}$ extended knowledge bases, where the size of each extended knowledge base is polynomial in m and exponential in n. Since the size of each concept that occurs in a conjunct of $\mathsf{cnf}(\mathsf{con}_{\mathcal{K}}(q))$ is polynomial in n and since these are the only concepts that can contain role conjunctions, the size of the longest role conjunction is also bounded by n. □

5.3 Deciding Query Entailment for $\mathcal{SHOQ}$

We now present our automata based decision procedure for $\mathcal{SHOQ}^{\sqcap}$ knowledge base consistency. By Theorem 5.14 this also gives us a decision procedure for unions of Boolean conjunctive queries in $\mathcal{SHOQ}$.

Since automata cannot directly handle transitive roles, we first transform a $\mathcal{SHOQ}^{\sqcap}$ knowledge base $\mathcal{K}$ into an equisatisfiable $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base $\mathcal{K}'$, i.e., in $\mathcal{K}'$ all roles are treated as non-transitive. We show that each model of $\mathcal{K}'$ in which we transitively close the interpretation of the roles that are transitive in $\mathcal{K}$ is a model of $\mathcal{K}$. The decision procedure for consistency of $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases that we present runs in single exponential deterministic time in the size of the input knowledge base. We use alternating automata, which have

already been used for obtaining worst-case optimal decision procedures for the hybrid μ-calculus [113] and for $\mathcal{ALCQIb}_{reg}$ [25, 30], which can encode $\mathcal{SHIQ}$ knowledge bases.

5.3.1 Canonical Models of Bounded Branching Degree

Since automata also rely on the tree/forest model property of the logic, we extend the definition of canonical models for a $\mathcal{SHOQ}$ knowledge base in the straightforward way to canonical models for $\mathcal{SHOQ}^{\sqcap}$ knowledge bases (cf. Definition 5.1 on page 122). Usually, automata work on trees of bounded branching degree, where the *branching degree* $d(w)$ of a node w in a tree T is the number of successors of w. If there is a k such that $d(w) \leq k$ for each $w \in T$, then we say that T has branching degree k. We now show that a consistent $\mathcal{SHOQ}^{\sqcap}$ knowledge base has a canonical model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where, for each $o \in \mathsf{nom}(\mathcal{K})$, the branching degree of the tree $\{w \mid (o, w) \in \Delta^{\mathcal{I}}\}$ is bounded by some k that is polynomial in the size of $\mathcal{K}$ assuming unary coding of numbers. For such a canonical model $\mathcal{I}$, we say that $\mathcal{I}$ has branching degree k. We use this result when we show how to eliminate transitivity, and when we introduce the abstractions of models that are accepted by our automata.

Definition 5.16. For a $\mathcal{SHOQ}^{\sqcap}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R})$, we define the *closure* $\mathsf{cl}(\mathcal{K})$ as the smallest set satisfying the following conditions:

- if $C \sqsubseteq D \in \mathcal{T}$, then $\mathsf{nnf}(\neg C) \sqcup D \in \mathsf{cl}(\mathcal{K})$,
- if D is a sub-concept of $C \in \mathsf{cl}(\mathcal{K})$, then $D \in \mathsf{cl}(\mathcal{K})$,
- if $D \in \mathsf{cl}(\mathcal{K})$, then $\mathsf{nnf}(\neg D) \in \mathsf{cl}(\mathcal{K})$.

△

Lemma 5.17. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base with $|\mathcal{K}| = m$, n_{max} the maximal number occurring in number restrictions, and $k = m \cdot n_{max}$. If $\mathcal{K}$ is consistent, then $\mathcal{K}$ has a canonical model $\mathcal{I}$ with branching degree k.*

Proof. Let $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be an interpretation such that $\mathcal{I}' \models \mathcal{K}$. From $\mathcal{I}'$, we construct a canonical model $\mathcal{I}$ for $\mathcal{K}$ with branching degree k and its forest base $\mathcal{J}$ as follows: we define a non-deterministic function choose that returns, for a concept $C \in \mathsf{cl}(\mathcal{K})$ and an element $d \in \Delta^{\mathcal{I}'}$ a subset $S_{C,d}$ of $\Delta^{\mathcal{I}'}$ such that

1. if $C = (\exists r.D)$ and $d \in C^{\mathcal{I}'}$, then $S_{C,d} = \{d'\}$, for some d' such that $(d, d') \in r^{\mathcal{I}'}$ and $d' \in D^{\mathcal{I}'}$,
2. if $C = (\geqslant n\ r.D)$ and $d \in C^{\mathcal{I}'}$, then $S_{C,d} = \{d_1, \ldots, d_n\}$, $(d, d_i) \in r^{\mathcal{I}'}$, and $d_i \in D^{\mathcal{I}'}$ for each i with $1 \leq i \leq n$,
3. $S_{C,d} = \emptyset$ otherwise.

We define the set $P \subseteq (\Delta^{\mathcal{I}'})^*$ of *paths* to be the smallest set such that

- for all $o \in \mathsf{nom}(\mathcal{K}), o^{\mathcal{I}'}$ is a path;
- $d_1 \cdots d_n \cdot d$ is a path, if
  - $d_1 \cdots d_n$ is a path,
  - $d_n \in C^{\mathcal{I}}$ for some $C \in \mathsf{cl}(\mathcal{K})$ with $d \in \mathsf{choose}(C, d_n)$,
  - if there is no $o \in \mathsf{nom}(\mathcal{K})$ such that $d = o^{\mathcal{I}'}$.

Now fix a set $S \subseteq \mathsf{nom}(\mathcal{K}) \times \mathbb{N}^*$ and a bijection $f\colon S \to P$ such that, for each $o \in \mathsf{nom}(\mathcal{K})$,

(i) $(o, \varepsilon) \in S$,

(ii) $\{w \mid (o, w) \in S\}$ is a tree,

(iii) $f(o, \varepsilon) = o^{\mathcal{I}'}$, and

(iv) if $(o, w), (o, w') \in S$ with w' a successor of w, then $f(o, w') = f(o, w) \cdot d$ for some $d \in \Delta^{\mathcal{I}'}$.

Due to the definition of choose, there are, for each path $\vec{d} \in P$, at most $m \cdot n_{max}$ elements $d_1, \ldots, d_{m \cdot n_{max}}$ such that $\vec{d} \cdot d_i$ with $1 \leq i \leq m \cdot n_{max}$ is a path in P. Since, additionally, f is a bijection, the tree $T = \{w \mid (o, w) \in S\}$ has branching degree k.

For all $(o, w) \in S$, set $\mathsf{Tail}(o, w) = d_n$ if $f(o, w) = d_1 \cdots d_n$. Now, define a forest base $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ for $\mathcal{K}$ as follows:

(a) $\Delta^{\mathcal{J}} = S$;

(b) for each $o \in \mathsf{nom}(\mathcal{K})$, $o^{\mathcal{J}} = (o, \varepsilon) \in S$;

(c) for each $C \in N_C, (o, w) \in C^{\mathcal{J}}$ iff $(o, w) \in S$ and $\mathsf{Tail}(o, w) \in C^{\mathcal{I}'}$;

(d) for each $r \in N_R$, $((o,w),(o',w')) \in r^{\mathcal{J}}$ iff either

(I) $w' = \varepsilon$ and $(\mathsf{Tail}(o,w), o'^{\mathcal{I}'}) \in r^{\mathcal{I}'}$ or

(II) $o = o'$, w' is a successor of w and $(\mathsf{Tail}(o,w), \mathsf{Tail}(o',w')) \in r^{\mathcal{I}'}$.

It is clear that $\mathcal{J}$ is a forest base for $\mathcal{K}$ due to the definition of S and the construction of $\mathcal{J}$ from S.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that is identical to $\mathcal{J}$ except that, for all non-simple roles r, we set

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq^*_{\mathcal{R}} r,\, s \in N_{tR}} (s^{\mathcal{J}})^+$$

We can prove that $\mathcal{I} \models \mathcal{K}$ inductively over the structure of concepts following the argumentation given in the proof of Lemma 4.2. Since the requirement on the branching degree is fulfilled as argued above, this proves the claim. □

5.3.2 Eliminating Transitivity

Transitivity axioms are, in general, known to be difficult to handle. Testing the consistency of a knowledge base in a logic with transitive roles can, however, often be polynomially reduced to checking an extended knowledge base without transitive roles [74, 78, 91]. We introduce this reduction first for $\mathcal{SHIQ}$ (a similar reduction works for $\mathcal{SHOIQ}$) and then show which difficulties arise when, instead of inverses, we have nominals plus role conjunctions, and how we can, nevertheless, eliminate transitivity. The price we have to pay, in particular for allowing both role conjunctions and transitivity, is that the translation is no longer polynomial, but exponential in the size of the longest role conjunction. Please note that this is also the case for the translation from $\mathcal{SHIQ}^{\sqcap}$ to $\mathcal{ALCQIb}$ in the previous chapter. To the best of our knowledge, it is unknown if this blow-up can be avoided.

Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base. The *closure* $\mathsf{cl}(\mathcal{K})$ of a $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ is defined as the closure for $\mathcal{SHOQ}^{\sqcap}$ KBs (cf. Definition 5.16), but, additionally, satisfies the following conditions:

- if $C(a) \in \mathcal{A}$, then $C \in \mathsf{cl}(\mathcal{K})$,
- if $\forall r.D \in \mathsf{cl}(\mathcal{K})$, $s \sqsubseteq^*_{\mathcal{R}} r$, and $s \in \mathsf{Trans}_{\mathcal{R}}$, then $\forall s.(\forall s.D) \in \mathsf{cl}(\mathcal{K})$.

Let $\mathsf{elimTrans}(\mathcal{K})$ be the $\mathcal{ALCHIQ}$ knowledge base obtained from $\mathcal{K}$ by treating all transitive roles as non-transitive and by adding an axiom $\forall r.C \sqsubseteq \forall t.(\forall t.C)$ for each concept $\forall r.C \in \mathsf{cl}(\mathcal{K})$ and role t such that $t \sqsubseteq^*_{\mathcal{R}} r$ and $t \in \mathsf{Trans}_{\mathcal{R}}$. Then $\mathsf{elimTrans}(\mathcal{K})$ is consistent iff $\mathcal{K}$ is consistent [91, Thm. 5.2.3] and the size of $\mathsf{elimTrans}(\mathcal{K})$ is polynomial in the size of $\mathcal{K}$.

The simultaneous presence of transitive roles, role conjunctions, and nominals makes the elimination of transitive roles more involved than usual [74, 78, 91]. For example, let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base with

$$\begin{aligned}\mathcal{T} = \{&\{o\} \sqsubseteq \exists t.A,\\ &A \sqsubseteq \exists t.B,\\ &B \sqsubseteq \exists t.(\{o'\}),\\ &\{o\} \sqsubseteq \exists r.(\{o'\})\},\end{aligned}$$

$\mathcal{R} = \emptyset$, and t a transitive role. Figure 5.7 shows a representation of a model for $\mathcal{K}$.

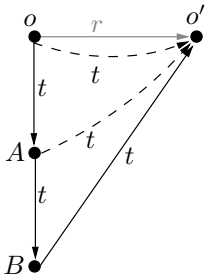

Figure 5.7: A representation of a model for $\mathcal{K}$, where the black edges represent the role t, the grey edge represents the role r and the dashed edges indicate shortcuts due to t being transitive.

It is not hard to check that adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ makes the knowledge base inconsistent.

We now introduce a naive extension of the function $\mathsf{elimTrans}$ to $\mathcal{SHOQ}^{\sqcap}$ input knowledge bases and show why this does not yield an equisatisfiable knowledge base as desired. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base. The function $\mathsf{elimTrans}(\mathcal{K})$ yields an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base obtained from $\mathcal{K}$ by treating all transitive roles as non-transitive and by adding an axiom

$$\forall(r_1 \sqcap \ldots \sqcap r_n).C \sqsubseteq \forall(t_1 \sqcap \ldots \sqcap t_n).(\forall(t_1 \sqcap \ldots \sqcap t_n).C)$$

for each concept $\forall(r_1 \sqcap \ldots \sqcap r_n).C \in \mathsf{cl}(\mathcal{K})$ and roles $t_1, \ldots, t_n \in N_{tR}$ such that $t_i \sqsubseteq^*_{\mathcal{R}} r_i$ for each i with $1 \leq i \leq n$.

For the above example knowledge base (cf. Figure 5.7), $\mathsf{elimTrans}(\mathcal{K})$ contains the additional axioms $\forall t.X \sqsubseteq \forall t.(\forall t.X)$ with $X \in \{A, B, \{o'\}\}$ and all roles are non-transitive. Adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ to $\mathcal{K}$ does not yield any additional axioms in the translation, since r is a simple role. Since none of the added axioms explicates the implicit t relation between the nominals o and o', extending $\mathcal{K}$ with the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ yields a consistent knowledge base after applying the translation, contradicting our assumption that a knowledge base $\mathcal{K}$ is consistent iff $\mathsf{elimTrans}(\mathcal{K})$ is consistent.

Intuitively, this problem arises since, even in canonical models, we can have arbitrary relations between nominals, and not only tree-like structures as in the remaining parts. This can lead to situations where, as in the above example, we have an explicit relationship between two nominals, but only together with the implicit transitive shortcut can the universal quantifier over the role conjunction be applied. To handle these situations correctly, we explicate all transitive shortcuts between nominals.

Definition 5.18. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base. The function $\mathsf{elimTrans}(\mathcal{K})$ yields the $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base obtained from $\mathcal{K}$ as follows:

1. regard all transitive roles as non-transitive,
2. for each role $t \in N_{tR}$ and each nominal $o \in \mathsf{nom}(\mathcal{K})$, add an axiom $\exists t.(\exists t.(\{o\})) \sqsubseteq \exists t.(\{o\})$, and
3. for each concept $\forall R.C \in \mathsf{cl}(\mathcal{K})$ with $R = r_1 \sqcap \ldots \sqcap r_n$ and roles $t_1, \ldots, t_n \in N_{tR}$ such that $t_i \sqsubseteq^*_{\mathcal{R}} r_i$ for each $1 \leq i \leq n$, add an axiom $\forall R.C \sqsubseteq \forall T.(\forall T.C)$, where $T = t_1 \sqcap \ldots \sqcap t_n$.

△

With the above definition, the resulting knowledge base $\mathsf{elimTrans}(\mathcal{K})$ for the example knowledge base $\mathcal{K}$, contains, additionally, the axioms $\exists t.(\exists t.(\{o\})) \sqsubseteq \exists t.(\{o\})$ and $\exists t.(\exists t.(\{o'\})) \sqsubseteq \exists t.(\{o'\})$. The latter one ensures that the implicit t-edges that are shown as dashed lines in Figure 5.7 are made explicit. As a consequence, adding the axiom $\{o\} \sqsubseteq \forall(r \sqcap t).(\neg\{o'\})$ indeed results in an inconsistent knowledge base as desired.

Lemma 5.19. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base. Then $\mathcal{K}$ is consistent iff* $\mathsf{elimTrans}(\mathcal{K})$ *is consistent.*

Proof. Let $R = r_1 \sqcap \ldots \sqcap r_n$ and $T = t_1 \sqcap \ldots \sqcap t_n$.

We start with the only if direction since it is rather trivial. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of $\mathcal{K}$. Clearly $\mathcal{I}$ is also a model of the knowledge base obtained by treating all roles as non-transitive. We show that $\mathcal{I}$ also satisfies the additional axioms. Assume that there is a role $t \in N_{tR}$ and a nominal o such that $d \in (\exists t.(\exists t.(\{o\})))^{\mathcal{I}}$. We have to show that this implies that $d \in \exists t.(\{o\})$. By definition of the semantics, $d \in (\exists t.(\exists t.(\{o\})))^{\mathcal{I}}$ implies that there are elements d' and d_o such that $(d, d'), (d', d_o) \in t^{\mathcal{I}}$ and $d_o = o^{\mathcal{I}}$. Transitivity of t then implies that $(d, d_o) \in t^{\mathcal{I}}$ and, hence, $d \in (\exists t.(\{o\}))^{\mathcal{I}}$ as required.

For the remaining axioms, assume that there is an element $d \in \Delta^{\mathcal{I}}$ such that $d \in (\forall R.C)^{\mathcal{I}}$, but $d \notin (\forall T.(\forall T.C))^{\mathcal{I}}$. Hence, by definition of the semantics, $d \in (\exists T.(\exists T.(\neg C)))^{\mathcal{I}}$ and there are elements $d', d'' \in \Delta^{\mathcal{I}}$ such that $(d, d'), (d', d'') \in T^{\mathcal{I}}$ and $d'' \in (\neg C)^{\mathcal{I}}$. Since $t_i \in N_{tR}$, for all $1 \leq i \leq n$, we have that $(d, d'') \in T^{\mathcal{I}}$ and, since $t_i \sqsubseteq r_i$ for each $1 \leq i \leq n$, we also have that $(d, d'') \in R^{\mathcal{I}}$. Since $d'' \in (\neg C)^{\mathcal{I}}, d \in (\exists R.(\neg C))^{\mathcal{I}}$, contradicting our initial assumption that $d \in (\forall R.C)^{\mathcal{I}}$.

For the if direction: Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a canonical model of $\mathsf{elimTrans}(\mathcal{K})$, which exists due to Lemma 5.17. Since $\mathsf{elimTrans}(\mathcal{K})$ is an extension of $\mathcal{K}$ apart from the fact that all roles are treated as non-transitive, $\mathcal{I} \models \mathcal{K}$ if we treat also all roles in $\mathcal{K}$ as non-transitive. Let $\mathcal{I}'$ be the interpretation that is identical to $\mathcal{I}$ except that, for all non-simple roles r, we have

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq_{\mathcal{R}} r,\, s \in N_{tR}} (s^{\mathcal{J}})^{+}.$$

Please note that, by definition of $\mathsf{elimTrans}(\mathcal{K})$, forest bases, and canonical models, $\mathcal{I}$ is a forest base for $\mathcal{I}'$. We have to show that $\mathcal{I}' \models \mathcal{T}$. Most importantly, we have to show that if $d \in (\forall R.C)^{\mathcal{I}}$, then $d \in (\forall R.C)^{\mathcal{I}'}$ since these are the only concepts affected by transitively closing the interpretation of transitive roles (recall that roles in number restrictions are simple). More precisely, assume that there are two elements $(o, w), (o', w') \in \Delta^{\mathcal{I}}$ such that $(o, w) \in (\forall R.C)^{\mathcal{I}}, (o', w') \in (\neg C)^{\mathcal{I}}, ((o, w), (o', w')) \notin R^{\mathcal{I}}$, but $((o, w), (o', w')) \in R^{\mathcal{I}'}$. By definition of forest bases and canonical interpretations, there is, for each j with $1 \leq j \leq n$, at least one transitive sub-role $t_j \in N_{tR}$ such that $t_j \sqsubseteq_{\mathcal{R}} r_j$ and a sequence of elements $(o_1, w_1), \ldots, (o_m, w_m)$ with $m > 2, (o_1, w_1) = (o, w), (o_m, w_m) = (o', w')$ and, for

each i with $1 \leq i < m, ((o_i, w_i), (o_{i+1}, w_{i+1})) \in t^{\mathcal{I}} \setminus t^{\mathcal{I}'}$. If $o_i = o = o'$, and w_{i+1} is a successor of w_i for each i with $1 \leq i < m$, this is rather trivial: since $\mathsf{elimTrans}(\mathcal{K})$ contains an axiom $\forall R.C \sqsubseteq \forall T.(\forall T.C)$ we have that $(o', w') \in C^{\mathcal{I}}$, contradicting our assumption. Hence, we assume that, for some i with $1 \leq i \leq m$, $o_i \neq o$ or, for some i with $1 \leq i < m, w_{i+1}$ is not a successor of w_i. Due to the canonicity of $\mathcal{I}$, we have that, for some i with $1 \leq i \leq m, (o_i, w_i) = (o', \varepsilon)$. We distinguish two cases:

1. $i = m$ and $(o_i, w_i) = (o_m, w_m) = (o', w') = (o', \varepsilon)$ and
2. $i < m$.

For (1): Since $m > 2$, by definition of the semantics and of canonical models, we have that the pair (o_{m-2}, w_{m-2}) is in $(\exists t_j.(\exists t_j.(\{o'\})))^{\mathcal{I}}$ for each j with $1 \leq j \leq n$. Since $t_j \in N_{tR}, o' \in \mathsf{nom}(\mathcal{K})$, and by definition of $\mathsf{elimTrans}(\mathcal{K})$ we have that $\mathsf{elimTrans}(\mathcal{K})$ contains an axiom $\exists t_j.(\exists t_j.(\{o'\})) \sqsubseteq \exists t_j.(\{o'\})$, which implies that $(o_{m-2}, w_{m-2}) \in (\exists T.(\{o'\}))^{\mathcal{I}}$. Repeating this argument for the preceding elements on the path yields that $(o_1, w_1) = (o, w) \in (\exists T.(\{o'\}))^{\mathcal{I}}, ((o, w), (o', w')) \in T^{\mathcal{I}} \cap T^{\mathcal{I}'}$ and, since $t_j \sqsubseteq^*_{\mathcal{R}} r_j$ for each j with $1 \leq j \leq n$, we have that $((o, w), (o', w')) \in R^{\mathcal{I}} \cap R^{\mathcal{I}'}$. Hence, $(o', w') \in C^{\mathcal{I}}$, contradicting our assumption.

For (2): Let i be the maximal $i < m$ such that $(o_i, w_i) = (o', \varepsilon)$. Applying the same arguments as for (1), we have that $((o, w), (o', \varepsilon)) \in R^{\mathcal{I}} \cap R^{\mathcal{I}'}$. By the canonicity of $\mathcal{I}$, and since $i < m$, we have that, for each j with $1 \leq j \leq n$, there is a transitive role $t_j \in N_{tR}$ such that $t_j \sqsubseteq^*_{\mathcal{R}} r_j$ and, since $\mathsf{elimTrans}(\mathcal{K})$ contains the axiom $\forall R.C \sqsubseteq \forall T.(\forall T.C)$, we have that $(o', \varepsilon) \in \forall T.C^{\mathcal{I}}$. Since i is maximal and by the canonicity of $\mathcal{I}$, (o', w_{j+1}) is a successor of (o', w_j) for each j with $i \leq j < m$ and we can apply the same arguments as used for the first case. □

Lemma 5.20. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base with $|\mathcal{K}| = m$ and the length of the longest role conjunction occurring in $\mathcal{K}$ is n. Then there is a polynomial p such that $|\mathsf{elimTrans}(\mathcal{K})|$ is bounded by $2^{p(n) \cdot \log p(m)}$.*

Proof. Both the number of nominals and the number of transitive roles in $\mathcal{K}$ is clearly bounded by m. Hence, the number of axioms from step (2) is at most quadratic in m. The number of concepts of the form $D = \forall R.C \in \mathsf{cl}(\mathcal{K})$ with $R = r_1 \sqcap \ldots \sqcap r_\ell$ is clearly also bounded by m. For each role r_i occurring in R we have at most m choices of transitive roles. Since the longest role conjunction is of length n, each such concept D can give rise to at most m^n axioms of size linear in the size of D, which proves the claim. □

In the remainder, we assume without further notice that existential and universal restrictions in $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases are expressed using number restrictions. This is clearly without loss of generality since all roles in an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base are simple, and hence we can make use of the following equivalences: $\exists R.C \equiv \geqslant 1\ R.C$ and $\forall R.C \equiv \leqslant 0\ R.(\neg C)$.

5.3.3 Alternating Automata

Alternating automata on infinite trees are a generalisation of non-deterministic automata on infinite trees [130], and they allow for a very elegant description of decision procedures for very expressive logics. They were first introduced by Muller and Schupp [95]. In this section, we devise an alternating automaton that accepts exactly (abstractions) of models of $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases. Such automata have first been used in the context of modal logics [141] and have also been extended to the hybrid μ-calculus [113] (with converse programs), i.e., for deciding the consistency of $\mathcal{ALCIO}$ knowledge bases with a universal role and fixpoints. The latter approach, however, lacks support for qualified number restrictions and adding those would result in a logic that is no longer decidable in ExpTime [132]. Recently, alternating automata have also been used for answering regular path queries in $\mathcal{ALCQIb}_{reg}$, which are a generalisation of unions of conjunctive queries [30]. Both of the aforementioned approaches use two-way alternating automata that are ideally suited for logics that allow for inverse roles (converse programs in the μ-calculus). Since the logic we consider here does not support inverse roles, we choose the slightly simpler standard (one-way) alternating automata, where we can only move downwards in the input tree.

We use the same definition for trees as in Definition 2.7 (cf. page 39). Additionally, for w a node in a tree T, we define $w \cdot 0 = w$. A *labelled tree* over an alphabet Σ is a pair $(T, \mathcal{L})$, where T is a tree and $\mathcal{L}\colon T \to \Sigma$ maps each node in T to an element of Σ.

Alternating automata have the power of making both universal and existential choices. Informally, this means that in the transition function, we can create copies of the automaton, send them to successor nodes, and require that either some (existential) or all (universal) of them are accepted. We use, as usual, positive Boolean formulae as defined below in the specification of the transition function.

Definition 5.21. Let X be a set of atoms. The set $\mathcal{B}^+(X)$ of *positive Boolean formulae* is built over atoms from X, true, and false using only the connectives $\wedge$ and $\vee$. Let $X^\top$ be a subset of X. We say that $X^\top$ *satisfies* a formula $\phi \in \mathcal{B}^+(X)$ if assigning true to all atoms in $X^\top$ and false to all atoms in $X \setminus X^\top$ makes ϕ true.

Let $[k] = \{0, 1, \ldots, k\}$. An *alternating looping tree automaton* on k-ary Σ-labelled trees is a tuple $\mathbf{A} = (\Sigma, Q, \delta, q_0)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta \colon Q \times \Sigma \to \mathcal{B}^+([k] \times Q)$ is the transition function.

A *run* of $\mathbf{A}$ on a Σ-labelled k-ary tree $(T, \mathcal{L})$ is a $(T \times Q)$-labelled tree $(T_r, \mathcal{L}_r)$ that satisfies the following conditions:

- $\mathcal{L}_r(\varepsilon) = (\varepsilon, q_0)$,
- if $y \in T_r$ with $\mathcal{L}_r(y) = (x, q)$ and $\delta(q, \mathcal{L}(x)) = \phi$, then there is a (possibly empty) set $S \subseteq [k] \times Q$ that satisfies ϕ such that, for each $(c, q') \in S$, y has a successor $y \cdot i$ in T_r with $i \in \mathbb{N}$ and $\mathcal{L}_r(y \cdot i) = (x \cdot c, q')$.

An automaton $\mathbf{A}$ *accepts* an input tree T if there exists a run of $\mathbf{A}$ on T. The language accepted by $\mathbf{A}$, $\mathsf{lang}(\mathbf{A})$, is the set of all trees accepted by $\mathbf{A}$. △

The transition function maps a state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a positive Boolean formula over $[k] \times Q$. For example, let the transition function for the initial state and an input letter σ be $\delta(q_0, \sigma) = ((1, q_1) \vee (2, q_2)) \wedge (3, q_3)$. When the automaton is in state q_0 and reads a node x labelled with σ, then it sends either a copy of itself in state q_1 to its first successor $x \cdot 1$ or a copy in state q_2 to its second successor $x \cdot 2$ and it sends a copy of itself in state q_3 to the third successor $x \cdot 3$.

For alternating automata, the *non-emptiness problem* is the following: given an alternating automaton $\mathbf{A}$, is there a tree $(T, \mathcal{L})$ such that $\mathbf{A}$ has an accepting run on $(T, \mathcal{L})$? It is known that this problem is solvable in time that is exponential in the number of $\mathbf{A}$'s states [139].

Please note that, since we use looping automata, we do not impose any acceptance conditions and each run is accepting, i.e., we require only that the conditions imposed on a run are satisfied. Other existing automata based procedures for Description or Modal Logics use Büchi or parity acceptance conditions [30, 113, 141], usually because the logics allow for the transitive closure operator to be used on roles, which is not the case for $\mathcal{ALCHOQ}^\sqcap$.

5.3.4 Tree Relaxations

In this section, we show how we can obtain labelled k-ary trees from a model for an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base such that these trees can be used as input for our automaton. We make use of canonical models here since they already have a kind of forest shape, which is a good starting point for building tree representations. Since the labelled trees that an automaton takes as input cannot have labelled edges, we additionally store, in the label of a node, with which roles it is related to its predecessor. Unfortunately, this does not work for the nominal nodes since a nominal node can be the successor of arbitrary elements and does not necessarily have a unique predecessor. In a first step, we build, therefore, a *relaxation* for a canonical model where, for each relationship between a node and a nominal node, we create a dummy node that is a representative of the nominal node. The label of the representative node is the extension of the label for the nominal node with rep and the role names with which the node is related to the nominal. For a graphical illustration, we use again the knowledge base $\mathcal{K}$ from Example 5.2 on Page 123. More precisely, we use the equisatisfiable $\mathcal{ALCHOQ}^{\sqcap}$ version $\mathsf{elimTrans}(\mathcal{K})$, which also contains the axioms $\exists t.(\exists t.(\{o\})) \sqsubseteq \exists t.(\{o\})$ and $\exists t'.(\exists t'.(\{o\})) \sqsubseteq \exists t'.(\{o\})$.

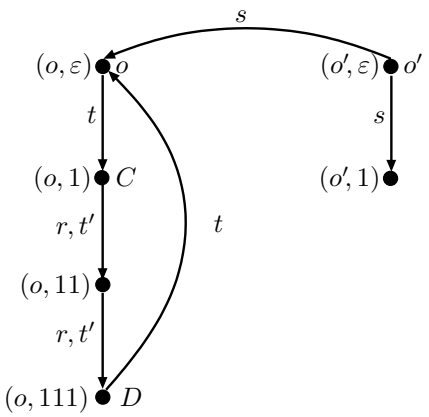

Figure 5.8: A representation of a canonical model $\mathcal{I}$ for $\mathsf{elimTrans}(\mathcal{K})$.

Figure 5.8 shows a canonical model for $\mathsf{elimTrans}(\mathcal{K})$ and Figure 5.9 shows a relaxation for $\mathcal{K}$ built from $\mathcal{I}$. In a second step, we build labelled trees from a relaxation, which we call *tree relaxations*.

Definition 5.22. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base over a signature $\mathcal{S} = (N_C, N_R, N_I)$. A set $H \subseteq \mathsf{cl}(\mathcal{K})$ is called a *Hintikka set* for $\mathcal{K}$ if the following conditions are satisfied:

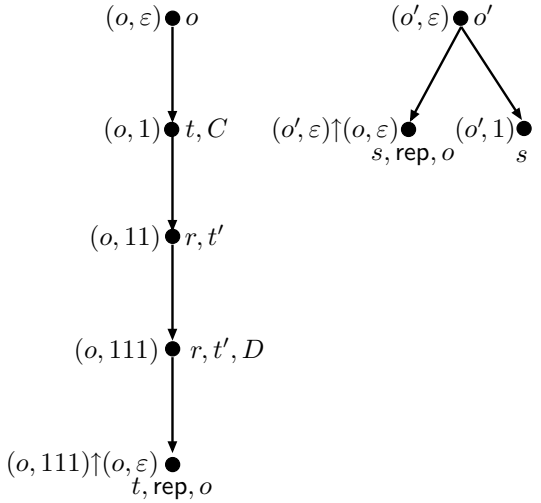

Figure 5.9: A graphical representation of a relaxation for $\mathcal{K}$.

1. For each $C \sqsubseteq D \in \mathcal{T}$, $\mathsf{nnf}(\neg C) \sqcup D \in H$.
2. If $C \sqcap D \in H$, then $\{C, D\} \subseteq H$.
3. If $C \sqcup D \in H$, then $\{C, D\} \cap H \neq \emptyset$.
4. There is no concept name $A \in N_C \cap \mathsf{cl}(\mathcal{K})$ with $\{A, \neg A\} \subseteq H$.
5. For all $C \in \mathsf{cl}(\mathcal{K})$, $C \in H$ or $\mathsf{nnf}(\neg C) \in H$.

We use $H(\mathcal{K})$ to denote the set of all Hintikka sets for $\mathcal{K}$.

A *relaxation* $R = (\Delta^{\mathcal{I}}, \mathcal{L})$ for $\mathcal{K}$ with $\mathcal{L}\colon \Delta^{\mathcal{I}} \to 2^{\mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\mathsf{rep}\}}$ satisfies the following properties:

(R1) Let $D = \mathsf{nom}(\mathcal{K}) \times \mathbb{N}^*$ and $B = \{d\!\uparrow\!d' \mid d \in D \text{ and } d' \in \mathsf{nom}(\mathcal{K}) \times \{\varepsilon\}\}$, then $\Delta^{\mathcal{I}} \subseteq D \cup B$.

(R2) For each $o \in \mathsf{nom}(\mathcal{K})$, $(o, \varepsilon) \in \Delta^{\mathcal{I}}$.

(R3) Each set $\{w \mid (o, w) \in \Delta^{\mathcal{I}} \cap D \text{ is a tree}\}$.

(R4) If $d\!\uparrow\!d' \in \Delta^{\mathcal{I}} \cap B$, then $\{d, d'\} \subseteq \Delta^{\mathcal{I}}, \mathcal{L}(d\!\uparrow\!d') \cap \mathsf{cl}(\mathcal{K}) = \mathcal{L}(d') \cap \mathsf{cl}(\mathcal{K})$, and $\mathsf{rep} \in \mathcal{L}(d\!\uparrow\!d')$.

(R5) For each $d \in \Delta^{\mathcal{I}}, \mathcal{L}(d) \cap \mathsf{cl}(\mathcal{K}) \in H(\mathcal{K})$.

(R6) For each $d \in \Delta^{\mathcal{I}}$, if $r \sqsubseteq s \in \mathcal{R}$ and $r \in \mathcal{L}(d)$, then $s \in \mathcal{L}(d)$.

(R7) For each $(o, \varepsilon) \in \Delta^{\mathcal{I}}, \mathcal{L}(o, \varepsilon) \cap \mathsf{rol}(\mathcal{K}) = \emptyset$.

(R8) If $d = (o, w) \in \Delta^{\mathcal{I}}$ and $(\geqslant n\ (r_1 \sqcap \ldots \sqcap r_k).C) \in \mathcal{L}(d)$, then there are n distinct elements $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$ such that, for each i with $1 \leq i \leq n, \{r_1, \ldots, r_k, C\} \subseteq \mathcal{L}(d_i)$ and either $d_i = (o, w \cdot c)$ with $c \in \mathbb{N}$ or $d_i = d{\uparrow} d' \in \Delta^{\mathcal{I}} \cap B$.

(R9) If $d = (o, w) \in \Delta^{\mathcal{I}}$ and $(\leqslant n\ (r_1 \sqcap \ldots \sqcap r_k).C) \in \mathcal{L}(d)$, then $\sharp(\{d' \in \Delta^{\mathcal{I}} \mid d' = (o, w \cdot c)$ for some $c \in \mathbb{N}$ or $d' = d{\uparrow} d_o \in \Delta^{\mathcal{I}} \cap B$ and $\{r_1, \ldots, r_k, C\} \subseteq \mathcal{L}(d')\}) \leq n$.

△

Lemma 5.23. *Let $\mathcal{K}$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base. $\mathcal{K}$ has a relaxation iff $\mathcal{K}$ is consistent.*

Proof. For the if direction, let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a canonical model of $\mathcal{K}$, which exists due to Lemma 5.17. From $\mathcal{I}$, we define a relaxation $R = (\Delta^{\mathcal{I}'}, \mathcal{L})$ for $\mathcal{K}$ where

$$\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{d{\uparrow} d' \mid d, d' \in \Delta^{\mathcal{I}} \text{ and there is some } r \in \mathsf{rol}(\mathcal{K}) \text{ and } o \in \mathsf{nom}(\mathcal{K}) \text{ such that } (d, d') \in r^{\mathcal{I}}, \text{ and } o^{\mathcal{I}} = d'\}$$

and $\mathcal{L}$ is the smallest set that satisfies the following conditions:

1. For each $d \in \Delta^{\mathcal{I}}, \{C \in \mathsf{cl}(\mathcal{K}) \mid d \in C^{\mathcal{I}}\} \subseteq \mathcal{L}(d)$.
2. For each $d{\uparrow} d' \in \Delta^{\mathcal{I}'} \setminus \Delta^{\mathcal{I}}, \{C \in \mathsf{cl}(\mathcal{K}) \mid d' \in C^{\mathcal{I}}\} \cup \{\mathsf{rep}\} \subseteq \mathcal{L}(d{\uparrow} d')$.
3. For each $(o, w) \in \Delta^{\mathcal{I}}$ with $w = w' \cdot c$ for $w' \in \mathbb{N}^*$ and $c \in \mathbb{N}, \{r \in \mathsf{rol}(\mathcal{K}) \mid ((o, w'), (o, w)) \in r^{\mathcal{I}}\} \subseteq \mathcal{L}(o, w)$.
4. For each $d{\uparrow} d' \in \Delta^{\mathcal{I}'} \setminus \Delta^{\mathcal{I}}, \{r \in \mathsf{rol}(\mathcal{K}) \mid (d, d') \in r^{\mathcal{I}}\} \subseteq \mathcal{L}(d{\uparrow} d')$.

It is not hard to check that, by definition of canonical models and by definition of R from $\mathcal{I}$, R is a relaxation for $\mathcal{K}$.

For the only if direction, let $R = (\Delta^{\mathcal{I}'}, \mathcal{L})$ be a relaxation for $\mathcal{K}$. We define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ from R as follows:

1. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'} \setminus \{d \mid \mathsf{rep} \in \mathcal{L}(d)\}$
2. For each $o \in \mathsf{nom}(\mathcal{K}), \{o\}^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \{o\} \in \mathcal{L}(d)\}$.

3. For each $A \in \mathsf{cl}(\mathcal{K}) \cap N_C$, $A^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid A \in \mathcal{L}(d)\}$.

4. For each $r \in \mathsf{rol}(\mathcal{K})$, $r^{\mathcal{I}} =$
$\{(d, d') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid r \in \mathcal{L}(d'), d = (o, w),$ and $d' = (o, w \cdot c)$ for $c \in \mathbb{N}\} \cup$
$\{(d, d') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid d{\uparrow}d' \in \Delta^{\mathcal{I}'}$ and $r \in \mathcal{L}(d{\uparrow}d')\}$

We first show that $\mathcal{I}$ is a model of $\mathcal{K}$. Clearly $\mathcal{I} \models \mathcal{R}$ due to Property R6 of relaxations and by definition of $\cdot^{\mathcal{I}}$ from R. Nominals are interpreted as singleton sets due to Properties R4, R2, and by definition of $\Delta^{\mathcal{I}}$ from $\Delta^{\mathcal{I}'}$. Since, by Property R5, $\mathcal{L}(d) \cap \mathsf{cl}(\mathcal{K})$ is a Hintikka set for each $d \in \Delta^{\mathcal{I}'}$, we only have to show that number restrictions are not violated. For each concept $C \in \mathsf{cl}(\mathcal{K})$, Condition 5 of Hintikka sets forces a decision whether C or $\mathsf{nnf}(\neg C)$ is in the set. This allows us, for an at least number restriction $\leqslant n\ (r_1 \sqcap \ldots \sqcap r_k).C$ to simply count the successors that contain $r_1, \ldots, r_k$ and C in their label. This, together with Properties R8, R9, and R1, and by definition of $\cdot^{\mathcal{I}}$ from R for roles proves the claim. □

In a second step, we build a so-called *tree relaxation* that is a labelled tree. For this, we additionally add a dummy root node labelled with root that has all nominal nodes as successors, and we require that the domain is a tree.

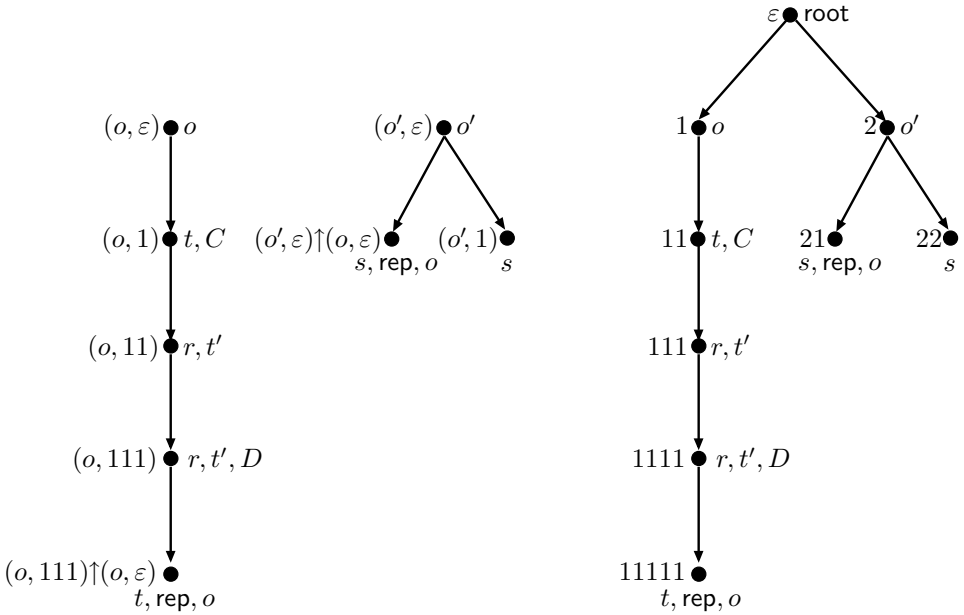

Figure 5.10: On the left hand side, we again see the relaxation from Figure 5.9 together with a tree relaxation built from it.

Figure 5.10 shows a representation of a tree relaxation built from the relaxation for our running example. The tree relaxation can, additionally, have dummy nodes labelled with #, but we do not show any dummy nodes in the figure. For ease of presentation, we assume in the remainder that all tree relaxations are full trees, i.e., all non-leaf nodes have the same number of successors, and we add dummy nodes labelled with # where necessary.

Definition 5.24. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base over a signature $\mathcal{S} = (N_C, N_R, N_I)$. A *tree relaxation* for $\mathcal{K}$ is a labelled tree $(T, \mathcal{L})$ with $\mathcal{L}\colon T \to 2^{\mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\mathsf{rep}, \#, \mathsf{root}\}}$ that satisfies the following conditions:

(T1) $\mathcal{L}(\varepsilon) = \{\mathsf{root}\}$ and, for each $w \in \mathbb{N}^+$, $\mathcal{L}(w) \cap \{\mathsf{root}\} = \emptyset$.

(T2) For each $o \in \mathsf{nom}(\mathcal{K})$, there is a unique $c \in \mathbb{N} \cap T$ with $o \in \mathcal{L}(c)$ and $\{\mathsf{rep}, \#, \mathsf{rol}(\mathcal{K})\} \cap \mathcal{L}(c) = \emptyset$.

(T3) If $c \in \mathbb{N} \cap T$ and $\mathsf{nom}(\mathcal{K}) \cap \mathcal{L}(c) = \emptyset$, then $\mathcal{L}(c) = \{\#\}$.

(T4) For each $w \in \mathbb{N}^+ \cap T$, $\sharp(\mathcal{L}(w) \cap \mathsf{nom}(\mathcal{K})) \leq 1$.

(T5) For each $w = w' \cdot c \in T$ with $w' \in \mathbb{N}^+$ and $c \in \mathbb{N}$, if $\mathcal{L}(w) \cap \mathsf{nom}(\mathcal{K}) \neq \emptyset$, then $\mathsf{rep} \in \mathcal{L}(w)$.

(T6) For each $w, w' \in T$ and $o \in \mathsf{nom}(\mathcal{K})$, if $o \in \mathcal{L}(w) \cap \mathcal{L}(w')$, then $\mathsf{cl}(\mathcal{K}) \cap \mathcal{L}(w) = \mathsf{cl}(\mathcal{K}) \cap \mathcal{L}(w')$.

(T7) For each $w \in \mathbb{N}^+ \cap T$, if $\{\mathsf{rep}, \#\} \cap \mathcal{L}(w) \neq \emptyset$, then, for each successor w' of w, $\# \in \mathcal{L}(w')$.

(T8) For each $w \in T$, if $\mathcal{L}(w) \cap \{\#, \mathsf{root}\} = \emptyset$, then $\mathcal{L}(w) \cap \mathsf{cl}(\mathcal{K}) \in H(\mathcal{K})$.

(T9) For each $w \in T$ and $r \sqsubseteq s \in \mathcal{R}$, if $r \in \mathcal{L}(w)$, then $s \in \mathcal{L}(w)$.

(T10) For each $w \in T$, if $(\geqslant n\ (r_1 \sqcap \ldots \sqcap r_m).C) \in \mathcal{L}(w)$, then there are at least n distinct successors $w_1, \ldots, w_n$ of w with $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, for each i with $1 \leq i \leq n$.

(T11) For each $w \in T$, if $(\leqslant n\ (r_1 \sqcap \ldots \sqcap r_m).C) \in \mathcal{L}(w)$, then there are at most n distinct successors $w_1, \ldots, w_n$ of w with $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, for each i with $1 \leq i \leq n$.

If T has branching degree k, then we say that $(T, \mathcal{L})$ is a *k-ary tree relaxation for* $\mathcal{K}$. △

Lemma 5.25. *Let $\mathcal{K}$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base, n_{max} the maximal number occurring in a number restriction in $\mathcal{K}$, and $k = n_{max} \cdot |\mathcal{K}| + \sharp(\mathsf{nom}(\mathcal{K}))$. $\mathcal{K}$ has a k-ary tree relaxation iff $\mathcal{K}$ is consistent.*

Proof. For the only if direction, we show that we can build a relaxation $R = (\Delta^{\mathcal{I}}, \mathcal{L}')$ for $\mathcal{K}$ from a k-ary tree relaxation $(T, \mathcal{L})$, which, by Lemma 5.23, is enough to prove the claim. We define a non-deterministic function choose that returns, for a concept $C \in \mathsf{cl}(\mathcal{K})$ and an element $w \in T$ such that $C \in \mathcal{L}(w)$ and $\mathsf{rep} \notin \mathcal{L}(w)$, a subset $S_{C,w}$ of T such that

1. if $C = (\geqslant n\ (r_1 \sqcap \ldots \sqcap r_m).D)$, then $S_{C,w} = \{w \cdot c_1, \ldots, w \cdot c_n\}$, where, for each i with $1 \leq i \leq m$, $c_i \in \{1, \ldots, k\}, \{r_1, \ldots, r_m, D\} \subseteq \mathcal{L}(w \cdot c_i)$, and $\# \notin \mathcal{L}(w \cdot c_i)$, and
2. $S_{C,w} = \emptyset$ otherwise.

Such a function exists by Property T10 of tree relaxations.

We now inductively define the domain $\Delta^{\mathcal{I}}$ of R together with a total and injective mapping $\tau \colon \Delta^{\mathcal{I}} \to T$. We first fix the roots of R: for each $o \in \mathsf{nom}(\mathcal{K})$, $(o, \varepsilon) \in \Delta^{\mathcal{I}}$ and $\tau(o, \varepsilon) = c$ for $c \in \{1, \ldots, k\}, o \in \mathcal{L}(c)$, and $\# \notin \mathcal{L}(c)$. Such a unique element c exists due to Property T2 of tree relaxations. For the induction: Let $(o, w) \in \Delta^{\mathcal{I}}, \tau(o, w) = \bar{w}, C = (\geqslant n\ R.D) \in \mathcal{L}(\bar{w}), c \in \mathsf{choose}(C, \bar{w})$. We distinguish two situations:

1. If $\mathsf{rep} \notin \mathcal{L}(\bar{w} \cdot c)$, then $(o, w \cdot c) \in \Delta^{\mathcal{I}}$ and $\tau(o, w \cdot c) = \bar{w} \cdot c$.
2. If $\mathsf{rep} \in \mathcal{L}(\bar{w} \cdot c)$, then $d{\uparrow}d' \in \Delta^{\mathcal{I}}$ with $d = (o, w), d' = (o', \varepsilon)$ for $o' \in \mathcal{L}(\bar{w} \cdot c)$, and $\tau(d{\uparrow}d') = \bar{w} \cdot c$. Such o' exists and is unique due to Properties T4 and T5.

The latter case adds those elements that represent links back to some nominal. Such elements have a different form from the other elements of the domain and have only dummy descendants in T due to Property T7.

Now let $\mathcal{L}' \colon d \mapsto \mathcal{L}(\tau(d))$, for each $d \in \Delta^{\mathcal{I}}$. It is not hard to check that R is indeed a relaxation for $\mathcal{K}$.

For the if direction: Let $k' = n_{max} \cdot |\mathcal{K}|$. By Lemma 5.17, $\mathcal{K}$ has a k'-ary canonical model $\mathcal{I}$. Let $(\Delta^{\mathcal{I}}, \mathcal{L}')$ be the relaxation built from $\mathcal{I}$ as shown in the

proof of Lemma 5.23. We use $(\Delta^{\mathcal{I}}, \mathcal{L}')$ to inductively define a tree relaxation $(T, \mathcal{L})$ for $\mathcal{K}$ together with a total and injective mapping τ from $\Delta^{\mathcal{I}}$ to T as follows:

1. For each $(o, \varepsilon) \in \Delta^{\mathcal{I}}$, there is a unique $c \in T \cap \mathbb{N}$ with $\tau(o, \varepsilon) = c$.
2. For each $(o, w), (o, w') \in \Delta^{\mathcal{I}}$, if w' is a successor of w, then $\tau(o, \varepsilon) \cdot w' \in T$ and $\tau(o, w') = \tau(o, \varepsilon) \cdot w'$.
3. For each $d{\uparrow}d' \in \Delta^{\mathcal{I}}$ with $\tau(d) = w$ and $\tau(d') = c, w' = w \cdot (k' + c) \in T$ and $\tau(d{\uparrow}d') = w'$.
4. For each $d \in \Delta^{\mathcal{I}}$, $\mathcal{L}(\tau(d)) = \mathcal{L}'(d)$.
5. $\mathcal{L}(\varepsilon) = \{\mathsf{root}\}$.

It is not hard to check that, for each $(o, w) \in \Delta^{\mathcal{I}}$, there are at most k' elements $(o, w') \in \Delta^{\mathcal{I}}$ such that w' is a successor of w and there are at most $\sharp(\mathsf{nom}(\mathcal{K}))$ elements of the form $(o, w){\uparrow}(o', \varepsilon) \in \Delta^{\mathcal{I}}$. Hence, the tree T has indeed branching degree k since, for each node, we require at most k' successors in Step 2 and at most $\sharp(\mathsf{nom}(\mathcal{K}))$ successors in Step 3. It is not hard to check that, by definition of $(T, \mathcal{L})$ from $(\Delta^{\mathcal{I}}, \mathcal{L}')$, $(T, \mathcal{L})$ satisfies all properties for tree relaxations. □

5.3.5 Deciding Existence of Tree Relaxations

In order to decide consistency of an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base $\mathcal{K}$, it remains to devise a procedure that decides whether $\mathcal{K}$ has a tree relaxation. For this, we define an alternating automaton that accepts exactly the tree relaxations of $\mathcal{K}$. More precisely, we first define two alternating automata $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$, and then define an automaton $\mathcal{B}_{\mathcal{K}}$ as their intersection. Informally, the automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ just checks that the input tree has a structure as required whereas the automaton $\mathcal{A}_{\mathcal{K}}$ checks that the input is indeed a tree relaxation for $\mathcal{K}$. For alternating automata, intersection is simple: we introduce a new initial state q_0 and set the transition function for q_0 and each letter σ from the input alphabet Σ to $\delta(q_0, \sigma) = (0, q_{(0,1)}) \wedge (0, q_{(0,2)})$, where $q_{(0,1)}$ and $q_{(0,2)}$ are the initial states of $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$ respectively. The size of the resulting automaton is the sum of the sizes of $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$.

The automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ is relatively straightforward and helps to keep the definition of the automaton $\mathcal{A}_{\mathcal{K}}$, where we do the real work, transparent. Informally, it guarantees the following:

- We distinguish root (state q_r), nominal (state q_o), nominal representative (state q_{rep}), dummy (state $q_\#$), and normal nodes (state q_n).
- The label root is only found in the root node.
- The level one nodes are either "real" nominal nodes (i.e., they are not marked as representatives with rep) with exactly one nominal and no roles in their label, or they are dummy nodes labelled with # only.
- The level one nominal nodes have either normal, nominal representative, or dummy nodes as successors.
- Nominal representative nodes are marked with rep, and have exactly one nominal in their label.
- Dummy nodes have only dummy nodes as successors.

More precisely, let n_{max} be the maximal number occurring in number restrictions in $\mathcal{K}$, and $k = n_{max} \cdot |\mathsf{cl}(\mathcal{K})| + \sharp(\mathsf{nom}(\mathcal{K}))$. The alphabet Σ for both automata $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$ is

$$2^{\{\mathsf{rep},\#,\mathsf{root}\} \cup \mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \mathsf{nom}(\mathcal{K})}.$$

We define $\bar{\mathcal{A}}_{\mathcal{K}}$ as $(\Sigma, \{q_r, q_o, q_n, q_{\mathsf{rep}}, q_\#\}, \bar{\delta}, q_r)$. The transition function $\bar{\delta}$ for each

$\sigma \in \Sigma$ is as follows:

$$\bar{\delta}(q_r, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}(i, q_o) \vee (i, q_\#) & \text{if } \sigma = \{\mathsf{root}\} \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\bar{\delta}(q_o, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}((i, q_n) \vee (i, q_{\mathsf{rep}}) \vee (i, q_\#)) & \text{if } \sharp(\mathsf{nom}(\mathcal{K}) \cap \sigma) = 1 \text{ and} \\ & \{\mathsf{root}, \mathsf{rep}, \#, \mathsf{rol}(\mathcal{K})\} \cap \sigma = \emptyset \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\bar{\delta}(q_n, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}((i, q_n) \vee (i, q_{\mathsf{rep}}) \vee (i, q_\#)) & \text{if } \{\mathsf{root}, \mathsf{rep}, \#, \mathsf{nom}(\mathcal{K})\} \cap \sigma = \emptyset \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\bar{\delta}(q_{\mathsf{rep}}, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}(i, q_\#) & \text{if } \sharp(\mathsf{nom}(\mathcal{K}) \cap \sigma) = 1, \mathsf{rep} \in \sigma, \text{ and } \{\mathsf{root}, \#\} \cap \sigma = \emptyset \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\bar{\delta}(q_\#, \sigma) = \begin{cases} \bigwedge_{i=1}^{k}(i, q_\#) & \text{if } \{\#\} = \sigma \\ \mathsf{false} & \text{otherwise} \end{cases}$$

The automaton $\mathcal{A}_\mathcal{K}$ mainly checks the formulae occurring in the labels of the input. Hence, most of the states correspond to formulae in $\mathsf{cl}(\mathcal{K})$ and the transition function is more or less determined by the semantics. In the root node, we additionally make a non-deterministic choice, for each nominal and each atomic concept, whether the concept or its negation holds at the nominal node. This choice is propagated downwards in the tree in order to ensure that the nominal representatives agree with their corresponding real nominal nodes on all atomic concepts. This also enables us to simply count over the successors of a node for the qualified number restrictions. We propagate the concepts via a kind of universal role and we assume that u is a symbol that does not occur in $\mathsf{cl}(\mathcal{K})$ or $\mathsf{rol}(\mathcal{K})$. We define, therefore, the following set of auxiliary states

$$\begin{aligned} Q_{\mathsf{rep}} = &\{\neg\{o\} \sqcup A \mid o \in \mathsf{nom}(\mathcal{K}) \text{ and } A \in N_C \cap \mathsf{cl}(\mathcal{K})\} \cup \\ &\{\neg\{o\} \sqcup \neg A \mid o \in \mathsf{nom}(\mathcal{K}) \text{ and } A \in N_C \cap \mathsf{cl}(\mathcal{K})\}. \end{aligned}$$

We then define $\mathcal{A}_\mathcal{K}$ as (Σ, Q, δ, q_0), where q_0 is the initial state and the set Q of states is

$$\{q_0\} \cup \mathsf{cl}(\mathcal{K}) \cup \mathsf{rol}(\mathcal{K}) \cup \{\neg r \mid r \in \mathsf{rol}(\mathcal{K})\} \cup \{q_\mathcal{T}, q_\mathcal{R}\} \cup Q_{\mathsf{rep}} \cup \{\forall u.C \mid C \in Q_{\mathsf{rep}}\} \cup$$
$$\{\langle \bowtie nR.C, i, j\rangle \mid \bowtie \in \{\leqslant, \geqslant\}, \bowtie nR.C \in \mathsf{cl}(\mathcal{K}), \text{ and } 0 \leq i, j \leq k\},$$

States of the form $\langle \bowtie n\ R.C, i, j\rangle$ are used to check that the number restrictions are satisfied. We now give a definition for the transition function together with an explanation for each of the different types of states. For each $\sigma \in \Sigma$, the transition function δ is defined as follows:

The Root Node At the root node, we are in the initial state q_0 which has the following tasks: (a) we make the non-deterministic guesses for all atomic concepts, (b) we check that there is exactly one nominal node for each of the nominals in $\mathsf{nom}(\mathcal{K})$, and (c) we make sure that the GCIs and in $\mathcal{T}$ and the RIAs in $\mathcal{R}$ are satisfied in all non-dummy descendants. The automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ guarantees already that each non-dummy level one node has exactly one nominal in the label. Let $\ell = \sharp(\mathsf{nom}(\mathcal{K}))$.

$$\begin{aligned}\delta(q_0, \sigma) = \quad & \bigwedge_{A \in \sigma \cap N_C} \bigwedge_{i=1}^{\ell} ((0, \forall u.(\neg\{o_i\} \sqcup A)) \vee (0, \forall u.(\neg\{o_i\} \sqcup \neg A))) \wedge \\ & \bigwedge_{i=1}^{\ell} \bigvee_{j=1}^{k} (j, \{o_i\}) \wedge \bigwedge_{1 \le i < j \le k} \left(\bigwedge_{o \in \mathsf{nom}(\mathcal{K})} (i, \neg\{o\}) \vee (j, \neg\{o\}) \right) \\ & \bigwedge_{i=1}^{k} (i, q_{\mathcal{T}}) \wedge (i, q_{\mathcal{R}})\end{aligned}$$

Propagating the Non-deterministic Choice for the Atomic Concepts at Nominal Nodes Whenever we are in a state that is used to propagate information downwards through the whole tree via the "universal role" and we are not at a dummy node, we check that the required concept holds at the current node and also check all successors. More precisely, for each $C \in Q_{\mathsf{rep}}$,

$$\delta(\forall u.C, \sigma) = \begin{cases} (0, C) \wedge \bigwedge_{i=1}^{k} (i, \forall u.C) & \text{if } \#, \mathsf{root} \notin \sigma \\ \bigwedge_{i=1}^{k} (i, \forall u.C) & \text{if } \mathsf{root} \in \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

TBox and RBox Axioms All non-dummy descendants of the root nodes must satisfy the TBox and RBox axioms, which is guaranteed by the following

transitions:

$$\delta(q_{\mathcal{T}},\sigma) = \begin{cases} \bigwedge_{C \sqsubseteq D \in \mathcal{T}} (0, (\mathsf{nnf}(\neg C) \sqcup D)) \wedge \bigwedge_{i=1}^{k} (i, q_{\mathcal{T}}) & \text{if } \# \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

$$\delta(q_{\mathcal{R}},\sigma) = \begin{cases} \bigwedge_{r \sqsubseteq s \in \mathcal{R}} ((0, \neg r) \vee (0, s)) \wedge \bigwedge_{i=1}^{k} (i, q_{\mathcal{R}}) & \text{if } \# \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

Atomic Concepts and Roles The concepts that are used as states are inductively decomposed according to the semantics. We start by defining the base cases:

$$\text{For each } \alpha \in (N_C \cap \mathsf{cl}(\mathcal{K})) \cup \mathsf{rol}(\mathcal{K}) \cup \mathsf{nom}(\mathcal{K})$$

$$\delta(\alpha,\sigma) = \begin{cases} \mathsf{true} & \text{if } \alpha \in \sigma \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\delta(\neg\alpha,\sigma) = \begin{cases} \mathsf{true} & \text{if } \alpha \notin \sigma \\ \mathsf{false} & \text{otherwise} \end{cases}$$

Nominals Since we use constructors for nominals, they are not handled as atomic concepts:

$$\text{For each } o \in \mathsf{nom}(\mathcal{K}),$$

$$\begin{aligned} \delta(\{o\},\sigma) &= (0, o) \\ \delta(\neg\{o\},\sigma) &= (0, \neg o) \end{aligned}$$

Conjunction and Disjunction Conjunction and disjunction are handled in the straightforward way:

$$\text{For each } C_1 \sqcap C_2 \in \mathsf{cl}(\mathcal{K}), \delta(C_1 \sqcap C_2, \sigma) = (0, C_1) \wedge (0, C_2)$$

$$\text{For each } C_1 \sqcup C_2 \in \mathsf{cl}(\mathcal{K}), \delta(C_1 \sqcup C_2, \sigma) = (0, C_1) \vee (0, C_2)$$

Number Restrictions For number restrictions, we have to use a more sophisticated technique that involves states that count how many successors have been checked and how many of the checked ones fulfill the requirements of the number restriction. This technique was introduced by Calvanese et al. [25]. More

precisely, for each concept of the form $(\geqslant n\ R.C) \in \mathsf{cl}(\mathcal{K})$ with $R = r_1 \sqcap \ldots \sqcap r_m$,

$$\delta(\geqslant n\ R.C, \sigma) = \begin{cases} (0, \langle \geqslant n\ R.C, 0, 0\rangle) & \text{if } \mathsf{rep} \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

For $1 \leq i \leq k$ and $1 \leq j \leq n$

$$\begin{aligned} \delta(\langle \geqslant n\ R.C, i, j\rangle, \sigma) = {} & (((i, \neg r_1) \vee \ldots \vee (i, \neg r_m) \vee (i, \mathsf{nnf}(\neg C))) \wedge \\ & (0, \langle \geqslant n\ R.C, i+1, j\rangle)) \vee \\ & ((i, r_1) \wedge \ldots \wedge (i, r_m) \wedge (i, C) \wedge \\ & (0, \langle \geqslant n\ R.C, i+1, j+1\rangle)) \end{aligned}$$

For $1 \leq i \leq k$

$$\delta(\langle \geqslant n\ R.C, i, n\rangle, \sigma) = \mathsf{true}$$

For $1 \leq j < n$

$$\delta(\langle \geqslant n\ R.C, k, j\rangle, \sigma) = \mathsf{false}$$

Informally, we use the counter i to count how many of the k successors have already been checked and j is increased for each successor that fulfills the requirements of the number restriction. The atmost number restrictions are handled similarly:

$$\delta(\leqslant n\ R.C, \sigma) = \begin{cases} (0, \langle \leqslant n\ R.C, 0, 0\rangle) & \text{if } \mathsf{rep} \notin \sigma \\ \mathsf{true} & \text{otherwise} \end{cases}$$

For $1 \leq i \leq k$ and $1 \leq j \leq n$

$$\begin{aligned} \delta(\langle \leqslant n\ R.C, i, j\rangle, \sigma) = {} & (((i, \neg r_1) \vee \ldots \vee (i, \neg r_m) \vee (i, \mathsf{nnf}(\neg C))) \wedge \\ & (0, \langle \leqslant n\ R.C, i+1, j\rangle)) \vee \\ & ((i, r_1) \wedge \ldots \wedge (i, r_m) \wedge (i, C) \wedge \\ & (0, \langle \leqslant n\ R.C, i+1, j+1\rangle)) \end{aligned}$$

For $1 \leq i \leq k$

$$\delta(\langle \leqslant n\ R.C, i, n+1\rangle, \sigma) = \mathsf{false}$$

For $1 \leq j < n$

$$\delta(\langle \leqslant n\ R.C, k, j\rangle, \sigma) = \mathsf{true}$$

We can now check whether the language accepted by the automaton $\mathcal{B}_\mathcal{K}$ is empty, which is enough to decide consistency of $\mathcal{K}$ as shown by the following theorem:

Theorem 5.26. *Let $\mathcal{K}$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base and $\mathcal{B}_{\mathcal{K}}$ an alternating automaton as defined above. Then $\mathcal{K}$ is consistent iff the language accepted by $\mathcal{B}_{\mathcal{K}}$ is non-empty.*

Proof. The if direction is mainly a consequence of the definition of the transition function for $\bar{\mathcal{A}}_{\mathcal{K}}$ and $\mathcal{A}_{\mathcal{K}}$. We show that each tree $(T, \mathcal{L})$ accepted by $\mathcal{B}_{\mathcal{K}}$ is a tree relaxation for $\mathcal{K}$ by showing that it satisfies all the conditions for tree relaxations. By Lemma 5.25 this suffices to prove the claim.

Property T1 holds by definition of $\bar{\mathcal{A}}_{\mathcal{K}}$. Property T2 holds by definition of the transition function for the initial state of $\mathcal{A}_{\mathcal{K}}$. Property T3 holds due to the definition of the transition function for the states q_r, q_o, and $q_{\#}$. Properties T4 and T5 hold by definition of the transition function for q_o, q_n, and q_{rep} in $\bar{\mathcal{A}}_{\mathcal{K}}$. Property T6 holds due to the definition of the transition function for q_0 and the states of the form $\forall u.(\neg\{o\} \sqcup C)$. Property T7 holds by definition of the transition function for q_{rep} and $q_{\#}$ in $\bar{\mathcal{A}}_{\mathcal{K}}$. Property T8 and T9 hold due to the definition of the transition function for $q_0, q_{\mathcal{T}}$, and $q_{\mathcal{R}}$ in $\mathcal{A}_{\mathcal{K}}$. Finally, Property T10 and T11 are a consequence of the transition functions for number restrictions in $\mathcal{A}_{\mathcal{K}}$.

For the only if direction: We show that, if $\mathcal{K}$ is consistent, then there is a tree relaxation $(T, \mathcal{L})$ for $\mathcal{K}$ such that $\mathcal{B}_{\mathcal{K}}$ has an accepting run on $(T, \mathcal{L})$. Let n_{max} be the maximal number occurring in a number restriction in $\mathcal{K}$ and $k = n_{max} \cdot |\mathsf{cl}(\mathcal{K})| + \sharp(\mathsf{nom}(\mathcal{K}))$. By Lemma 5.25, $\mathcal{K}$ has a k-ary tree relaxation $(T, \mathcal{L})$.

We first show that $\bar{\mathcal{A}}_{\mathcal{K}}$ accepts $(T, \mathcal{L})$: due to Property T1, the root and only the root of T is labelled with $\{\mathsf{root}\}$ as required by the transition function for the initial state. Due to Property T3, all non-nominal successors are dummy nodes labelled with $\{\#\}$. Due to Property T2, the labels of all nominal successors (state q_o) contain exactly one nominal, no roles, and neither rep nor $\#$. All successors of the nominal nodes have to be normal nodes (state q_n), nominal representatives (state q_{rep}), or dummy nodes (state $q_{\#}$). For $q_{\#}$, this is quite trivial, due to Property T7. For q_{rep}, due to Properties T5 and T4, a successor has at most one nominal in its label and, if it has one, then it also has rep in its label. All other states are trivially normal states and the transition function is satisfied.

Hence, we have to show that $\mathcal{A}_{\mathcal{K}}$ also accepts $(T, \mathcal{L})$. First, due to Property T6, a nominal node and its representatives agree on all atomic concepts, so the non-deterministic choice for the atomic concepts and states q_{rep} can easily be satisfied. The other conditions for the initial state hold due to Properties T2, T3,

T8, and T9. The conditions for $q_{\mathcal{T}}$ and $q_{\mathcal{R}}$ hold due to Properties T8 and T9. The transition function on atomic concepts, roles, conjunction, and disjunction trivially hold since labels are Hintikka sets due to Property T8. Finally, the number restrictions are satisfied due to the definition of Hintikka sets and by Properties T10 and T11. Hence, $\mathcal{B}_{\mathcal{K}}$ accepts $(T, \mathcal{L})$ as required. □

5.3.6 Combined Complexity

As before and without loss of generality (cf. Section 2.4), we assume for the complexity analysis that all concepts occur in concept conjuncts of the input query are literals, i.e., concept names or negated concept names. We first analyse the complexity of checking the consistency of $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases. We can combine this result with the results from Lemma 5.19 and Lemma 5.20 about reducing consistency checking of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases to consistency checking of $\mathcal{ALCHOQ}^{\sqcap}$ knowledge bases.

Theorem 5.27. *Let $\mathcal{K}$ be an $\mathcal{ALCHOQ}^{\sqcap}$ knowledge base with $|\mathcal{K}| = m$, then deciding the consistency of $\mathcal{K}$ can be done in deterministic time in $2^{p(m)}$ for a polynomial p assuming unary encoding of numbers in number restrictions.*

Proof. Since looping alternating tree automata are a special case of alternating Büchi tree automata, we can use the result of Vardi [139] that, for an alternating Büchi automaton $\mathbf{A}$ with n states and input alphabet with ℓ elements, non-emptiness of the language accepted by $\mathbf{A}$ is decidable in time exponential in n and polynomial in ℓ.

The alphabet of $\mathcal{B}_{\mathcal{K}}$ is $2^{\{\mathsf{rep},\#,\mathsf{root}\}\cup\mathsf{cl}(\mathcal{K})\cup\mathsf{rol}(\mathcal{K})\cup\mathsf{nom}(\mathcal{K})}$ and, since $\mathsf{cl}(\mathcal{K})$, $\mathsf{rol}(\mathcal{K})$ and $\mathsf{nom}(\mathcal{K})$ are linear in m, the number of symbols in Σ is bounded by $2^{p(m)}$ for some polynomial p.

The automaton $\bar{\mathcal{A}}_{\mathcal{K}}$ consists of just 5 states. For the automaton $\mathcal{A}_{\mathcal{K}}$, the cardinality of the sets $\mathsf{cl}(\mathcal{K})$ and $\mathsf{rol}(\mathcal{K})$ is linear in m. The cardinality of Q_{rep} is at most quadratic in m. The number of states introduced for the number restrictions is polynomial in m (with unary coding of numbers) and, counting all states together, we get that their number is again polynomial in m. Hence, emptiness of $\mathcal{B}_{\mathcal{K}}$ can be decided in time single exponential in m. □

Since, by Lemma 5.19 and Lemma 5.20, a $\mathcal{SHOQ}^{\sqcap}$ knowledge base of size m whose longest role conjunction is of length n can be reduced to an equisatisfiable

$\mathcal{ALCHOQ}^{\sqcap}$ knowledge base of size polynomial in m and exponential in n, we obtain the following upper bound on deciding consistency of $\mathcal{SHOQ}^{\sqcap}$ knowledge bases.

Theorem 5.28. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}^{\sqcap}$ knowledge base where $|\mathcal{K}| = m$ and the length of the longest role conjunction occurring in $\mathcal{K}$ is n. Deciding the consistency of $\mathcal{K}$ can be done in deterministic time in $2^{p(m)\cdot 2^{p(n)}}$ for a polynomial p assuming unary encoding of numbers in number restrictions.*

A straightforward consequence of the above is that, without role conjunctions, i.e., for $\mathcal{SHOQ}$ knowledge bases, the problem is in ExpTime. A corresponding lower bound follows immediately from the hardness result for $\mathcal{ALC}$ with general TBoxes [116].

Corollary 5.29. *For a $\mathcal{SHOQ}$ knowledge base $\mathcal{K}$ with $|\mathcal{K}| = m$, deciding the consistency of $\mathcal{K}$ is an* ExpTime*-complete problem assuming unary encoding of numbers in number restrictions.*

Combining Theorem 5.14, Theorem 5.28, and the upper bounds on the number and sizes of extended knowledge bases from Lemma 5.15, we get the following:

Theorem 5.30. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base with $|\mathcal{K}| = m$ assuming unary coding of numbers and q a union of connected Boolean conjunctive queries with $|q| = n$. Deciding whether $\mathcal{K} \models q$ under the unique name assumption can be done in deterministic time in $2^{p(m)2^{p(n)}}$ for some polynomial p.*

We can apply the same technique (cf. Lemma 4.25 and Lemma 2.9) that we used for $\mathcal{SHIQ}$ in order to show that the complexity results carry over when we do not make the unique name assumption and/or when we do not require the disjuncts in the input query to be connected conjunctive queries, which gives the following corollary:

Corollary 5.31. *Let $\mathcal{K}$ be a $\mathcal{SHOQ}$ knowledge base with $|\mathcal{K}| = m$ assuming unary coding of numbers and q a union of Boolean conjunctive queries with $|q| = n$. Deciding whether $\mathcal{K} \models q$ can be done in deterministic time in $2^{p(m)2^{p(n)}}$ for some polynomial p.*

5.3.7 Consequential Results

Due to the correspondence between query containment and query answering [22], the algorithm can also be used to decide containment of two unions of conjunctive queries over a $\mathcal{SHOQ}$ knowledge base, which gives the following result:

Corollary 5.32. *Given a $\mathcal{SHOQ}$ knowledge base $\mathcal{K}$ and two unions of conjunctive queries q and q', the problem whether $\mathcal{K} \models q \subseteq q'$ is decidable.*

By using [106, Thm. 11], we also show that the consistency of a $\mathcal{SHOQ}$ knowledge base extended with (weakly-safe) Datalog rules is decidable.

Corollary 5.33. *The consistency of $\mathcal{SHOQ}$+log-KBs (both under FOL semantics and under NM semantics) is decidable.*

5.4 Summary

We have presented a decision procedure for unions of Boolean conjunctive queries in $\mathcal{SHOQ}$ that runs in deterministic time single exponential in the size of the input knowledge base and double exponential in the size of the input query. As for the complexity of knowledge base consistency, the upper bounds for $\mathcal{SHOQ}$ agree with the ones for $\mathcal{SHIQ}$. It is still an open problem as to whether this is also the case for the lower bounds. The hardness proof for $\mathcal{SHIQ}$ [88] very much relies on the ability to propagate information upwards via inverse roles. Nominals however, also give the possibility of propagating information upwards, which may be used to obtain a similar hardness result for $\mathcal{SHOQ}$. The complexity results for $\mathcal{SHOQ}$ are currently restricted to unary coding of numbers, but a similar construction as used by Tobies [132] could possibly be used to show that the result also applies to the case of binary coding of numbers.

The automata based decision procedure that we present decides the consistency of $\mathcal{SHOQ}$ knowledge bases in deterministic exponential time. A corresponding lower bound for the problem immediately follows from the ExpTime-hardness result for deciding the satisfiability of an $\mathcal{ALC}$-concept w.r.t. general TBoxes [116]. Hence, we show that knowledge base consistency in $\mathcal{SHOQ}$ is ExpTime-complete. For $\mathcal{SHOQ}^{\sqcap}$, our algorithm runs in deterministic time double exponential in the size of the input knowledge base and, as for $\mathcal{SHIQ}^{\sqcap}$, it is still an open question whether this blow-up due to role conjunctions over non-simple roles can be avoided.

The problem of data complexity for $\mathcal{SHOQ}$ is not as clearly defined as for $\mathcal{SHIQ}$, since, in the presence of nominals, we can internalise the ABox and the clear separation between schema and data is no longer given. One could, however, still separate the part of the data that uses nominals only in a form that corresponds to ABox assertions and study the complexity of the algorithm w.r.t. this part as input, but we leave the study of the data complexity of this problem for future work.

As for $\mathcal{SHIQ}$, it is worth noting that, except for the collapsing step, all rewriting steps are only applicable to non-simple roles. It is, therefore, not unlikely that most real-world queries will produce a relatively small set of rewritings. Since the size of the query is usually small compared to the size of the queried knowledge base, and since the algorithm is double exponential "only" in the size of the query, a more goal directed version of the algorithm might still be useful for practical applications.

Chapter 6

Conclusions

In this thesis, we have presented several results regarding conjunctive query entailment. We have pointed out mistakes and sources of incompleteness in existing algorithms, presented new decision procedures and query answering techniques together with a range of complexity results, which we briefly summarise in this chapter.

6.1 Thesis Achievements

In Chapter 3, we have introduced related work and analysed existing algorithms for conjunctive query entailment in expressive Description Logics. We analyse why the decision procedure for conjunctive query entailment in $\mathcal{DLR}_{reg}$ by Calvanese et al. [22] is incomplete when regular expressions are allowed in the queries.

Secondly, we briefly introduce the CARIN system [87], which provides a decision procedure for conjunctive query entailment in the Description Logic $\mathcal{ALCNR}$ and which has been extended to $\mathcal{SHIQ}$ for queries with only simple roles [97]. We give an example of why the completeness proof for the extension of the CARIN technique to $\mathcal{SHIQ}$ with arbitrary roles in the query [98] does not work and we discuss why a straightforward extension of this technique to $\mathcal{SHOIQ}$ with only simple roles in the query is either incomplete or non-terminating.

Thirdly, we show in this chapter how the $\downarrow$ operator known from Hybrid Logics can be used in the setting of query answering. This operator allows a very elegant way of expressing queries as concepts but, in an unrestricted form, it makes even the basic Description Logic $\mathcal{ALC}$ undecidable. We show, however,

that for deciding query entailment, we need only a very restricted form of the $\downarrow$ binder and we sketch how the tableau algorithm for $\mathcal{SHIQ}$ can be extended in order to deal with the concepts that we obtain from a conjunctive query.

The main results of this thesis are presented in Chapter 4 and in Chapter 5, where we present decision procedures for entailment of unions of conjunctive queries in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$, respectively. Prior to our work, it was open whether conjunctive query entailment was decidable in such expressive Description Logics. The presented algorithms rewrite a given query into a set of queries that are of a simpler structure. Using the standard rolling-up technique, we then express these simpler queries as concepts that may additionally use role conjunctions. The task of checking query entailment is then reduced to possibly several knowledge base consistency checks for $\mathcal{SHIQ}^{\sqcap}$ and $\mathcal{SHOQ}^{\sqcap}$, respectively. For checking knowledge base consistency in $\mathcal{SHIQ}^{\sqcap}$, we extend the translation given by Tobies [132], which allows us to exploit the complexity results obtained for $\mathcal{ALCQI}b$ in our setting. For $\mathcal{SHOQ}^{\sqcap}$, we reduce the problem of knowledge base consistency to the emptiness problem for looping tree automata. We further analyse the complexity of the given decision procedures and derive 2-ExpTime upper bounds on the combined complexity of the problem in both logics. For $\mathcal{SHIQ}$, a recent result on the lower bound of the problem [88] yields that conjunctive query entailment is 2-ExpTime-complete. Query answering for $\mathcal{SHIQ}$ is, therefore, strictly harder than instance retrieval. Regarding data complexity, we show that deciding entailment of unions of conjunctive queries is a co-NP-complete problem in $\mathcal{SHIQ}$. Hence, regarding data complexity, it is not harder than other standard reasoning tasks and, even for the much weaker DL $\mathcal{ALE}$, the standard reasoning tasks are co-NP-hard regarding data complexity.

6.2 Significance of the Results

With the growing number of Semantic Web related applications, the demand for automated reasoning facilities as provided by modern DL reasoners increases. Although DL reasoners provide highly optimised reasoning support for the standard reasoning tasks such as checking concept subsumption or knowledge base consistency, they only provide limited support for ABox query answering, typically only providing for the retrieval of the instances of a possibly complex concept. This is not surprising due to the lack of decision procedures for more expressive query

languages for the DLs that are underpinning the web ontology language OWL. Query answering is likely to become increasingly important, however, as ontologies mature and are more widely deployed in applications. This is also reflected by the work of the Data Access Working Group that developed the W3C recommendation SPARQL[1]: a proposed standard for a conjunctive query language that can be used with Semantic Web languages such as RDF and OWL.

In this thesis, we make a significant step towards more expressive query answering facilities for Semantic Web applications: we present a decision procedure for conjunctive query entailment in OWL Lite.[2] The main challenge for a decision procedure for the more expressive OWL DL is the additional support of nominals. In this direction we also make a step forward, by presenting a decision procedure for $\mathcal{SHOQ}$, which allows for nominals, but does not support inverse roles as would be required for OWL DL.

Finally, it is interesting to note that conjunctive queries also play an important rôle in rule language extensions for Description Logics [106]. The least restrictive but still decidable rule extension proposed so far is the $\mathcal{DL}$+*log* framework [106]. This extension is decidable for a given Description Logic $\mathcal{DL}$ iff the entailment problem for unions of conjunctive queries is decidable for that $\mathcal{DL}$. The results from Chapter 4 and 5 immediately imply, therefore, the decidability of this rule extension in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$.

We believe that the results presented in this thesis will be the basis of more expressive reasoning services and are, therefore, directly relevant for implementors of DL reasoners and, indirectly, for users of DL based systems. The analysis of what is difficult in devising decision procedures for query entailment in expressive DLs will also be helpful for other researchers who want to develop extensions of query answering algorithms by either allowing for more expressive logics or more expressive query languages.

Our complexity results show that, although $\mathcal{SHIQ}$ conjunctive query entailment is a reasoning task that is strictly harder than the standard $\mathcal{SHIQ}$ reasoning problems, the data complexity is the same. The data complexity is considered to be a more informative performance estimate whenever the size of the TBox and query is small compared to the size of the ABox, which is often the case in practical applications. A more detailed analysis of our algorithms also shows

[1] http://www.w3.org/TR/2007/PR-rdf-sparql-query-20071112/

[2] More precisely, in an extension of OWL Lite with qualified number restrictions.

that queries that require a large number of knowledge base consistency checks are relatively rare since most of the rewriting steps are only applicable to queries that use non-simple roles.

6.3 Future Work

The work presented in this thesis can, basically, be extended in two directions: one is to improve the practicability of the algorithms and the other is to extend the framework by allowing for a more expressive underlying logic or by allowing for a more expressive query language.

For improving practicability, there are a few optimisations that directly come to mind and we briefly describe some of the options here:

1. Currently, the algorithms are not goal directed at all. For example, the forest rewriting step (cf. Definition 4.5 on page 88) uses the number of atoms in the query as a bound on the number of atoms that can replace one role atom. The goal of this step is, however, to obtain a query that is forest-shaped, and we can reduce the number of rewritings to those that can indeed lead to a forest-shaped query.

2. A very simple and straightforward optimisation for the algorithms would be to roll up all non-cyclic parts of the query as a kind of preprocessing step. More precisely, we can, as a first step, replace all tree-shaped parts of the query with a concept atom. By Lemma 4.7 and Lemma 5.10, this does not affect the correctness of the algorithm. This clearly reduces the number of atoms and variables in the query and, therefore, the overall number of produced rewritings.

3. For cycles in a query that do contain a constant, we can use this constant or a representative concept for it, and express the cyclic sub-query as a concept in which we use the representative concept to enforce the necessary co-reference. As in the previous optimisation, this can be done before any query rewriting is applied and reduces, therefore, the overall number of rewritings and knowledge base consistency checks that are required.

4. Using a query entailment decision procedure for query answering is not very efficient since we need one query entailment check for each possible

answer tuple. For finding suitable candidate tuples we can use heuristics, i.e., simpler queries that help to exclude obvious non-candidates, or cached results of previous queries. Some work has already been carried out in this direction [41, 122, 143]. Returning "cheap" answers first, i.e., answers that can be found without involving complex reasoning, might also help in practical implementations. This technique is, for example, already successfully employed in the RACER system [143].

The above list is certainly not exhaustive, and many ideas from databases can also be incorporated in order to make conjunctive query answering a more feasible reasoning task in practise. The algorithms presented in this thesis are only the starting point for implementable systems; however, our detailed analysis of the models, in particular in the correctness proofs of the query rewriting steps, might also give valuable insights when starting to devise optimisations.

Regarding an extension to more expressive logics, $\mathcal{SHOIQ}$ is certainly one of the first logics that come to mind since it underlies OWL DL. It is, therefore, reasonable to ask, why we cannot simply combine the two algorithms that we have for $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ in order to devise a decision procedure for conjunctive query entailment in $\mathcal{SHOIQ}$. The logic $\mathcal{SHOIQ}$ is, however, "trickier" than $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$. The DL $\mathcal{SHOIQ}$ is not only in a higher complexity class (NEXPTIME-complete [132] instead of EXPTIME-complete as $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$), but also devising a decision procedure for knowledge base consistency proved to be a more difficult task and was a long-standing open problem. Most importantly, for $\mathcal{SHOIQ}$ we do not know how to fix the names of the nominals a priori. Recall that a very essential property used in our proofs is the fact that we can restrict our attention to the canonical models of a knowledge base. The domains of these canonical models consist of a collection of trees such that the roots correspond to the individuals occurring in the ABox of the $\mathcal{SHIQ}$ input knowledge base or the nominals in the input $\mathcal{SHOQ}$ knowledge base. In $\mathcal{SHOIQ}$, however, we cannot restrict our attention to canonical models in which only the nominals from the input knowledge base occur as roots of the trees. If our canonical models contain also roots that do not correspond to nominals from the input knowledge base and we use only the existing nominal names in the rewriting procedure, then our algorithm would be incomplete. On the other hand, we cannot just introduce arbitrary new nominal names. A similar problem already occurs when extending the query entailment algorithm from the CARIN

system to $\mathcal{SHOIQ}$ with queries that are restricted to contain only simple roles (cf. the discussion in Section 3.5.2).

Existing algorithms for $\mathcal{SHOIQ}$ knowledge base consistency [63, 78] might also introduce new nominals during the execution of the algorithm but, for knowledge base consistency, we just need to find one model and it is possible to prove an upper bound on the number of the required new nominals. For query entailment, however, proving the existence of a model for the knowledge base is not enough. We need to make a statement about every model of the knowledge base. Studying the model theoretic properties of $\mathcal{SHOIQ}$ or related logics such as the guarded fragment with counting or the two variable fragment with counting might help to understand the structure of canonical models better. This could help to devise a suitable adapted unravelling procedure for constructing canonical models and, consequently, a suitably extended decision procedure for $\mathcal{SHOIQ}$.

Since we reduce the problem of query entailment in $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ to knowledge base consistency in $\mathcal{SHIQ}^{\sqcap}$ and $\mathcal{SHOQ}^{\sqcap}$ respectively, analysing the problem of query entailment w.r.t. $\mathcal{SHIQ}^{\sqcap}$ and $\mathcal{SHOQ}^{\sqcap}$ knowledge bases, i.e., where we allow role conjunctions already in the input knowledge base, might also be interesting. Currently, our proofs do not cover this case, but an extension might be possible.

Bibliography

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the 17th ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS 1998)*, pages 254–263, 1998. [48]

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995. [48]

[3] A. V. Aho, Y. Sagiv, and J. D. Ullman. Efficient optimization of a class of relational expressions. *ACM Transactions on Database Systems*, 4:297–314, 1979. [48]

[4] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998. [66]

[5] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. Query answering in expressive variants of dl-lite. In *Proceedings of the 15th Italian Conference on Database Systems (SEBD 2007)*, pages 250–257, 2007. [45]

[6] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 446–451, Sydney, Australia, 1991. [34]

[7] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proceedings of the Workshop on Processing Declarative Knowledge (PDK 1991)*, number 567 in Lecture Notes in Artificial Intelligence, pages 67–86. Springer-Verlag, 1991. [14]

[8] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003. [12, 15]

[9] F. Baader, C. Lutz, and B. Suntisrivaraporn. Efficient reasoning in $\mathcal{EL}^+$. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, CEUR Workshop Proceedings, 2006. [14]

[10] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference. Technical report, World Wide Web Consortium, 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/. [15]

[11] T. Berners-Lee, M. Fischetti, and M. L. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, 1999. [15]

[12] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):34–43, 2001. [15]

[13] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Special issue on decompositions of first-order logic. [60, 62]

[14] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2004. Reprinted with corrections. [46]

[15] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996. [49]

[16] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnik. CLASSIC:a structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989. [14]

[17] R. J. Brachman. On the epistemological status of semantic networks. *Associative Networks*, 1975. [14]

[18] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI 1984)*, pages 34–37, 1984. [14]

[19] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985. [14]

[20] R. J. Brachman, V. Pigman Gilbert, and H. J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI 1985)*, pages 532–539, 1985. [14]

[21] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 298–302. IOS Press, 2004. [14]

[22] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS 1998)*, pages 149–158. ACM Press and Addison Wesley, 1998. [11, 21, 25, 42, 48, 49, 50, 62, 67, 70, 81, 84, 90, 118, 120, 163, 165]

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR 1998)*, 1998. [16, 48]

[24] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications (DEXA 1998)*, pages 192–197. IEEE Computer Society Press, 1998. [48]

[25] D. Calvanese, G. De Giacomo, and M. Lenzerini. 2ATAs make DLs easy. In *Proceedings of the 2002 Description Logic Workshop (DL 2002)*, volume 53. CEUR (`http://ceur-ws.org/`), 2002. [139, 158]

[26] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Dl-lite: Tractable description logics for ontologies. In M. M. Veloso and S. Kambhampati, editors, *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 602–607. AAAI Press/The MIT Press, 2005. [46]

[27] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270. AAAI Press/The MIT Press, 2006. [45]

[28] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007. [45]

[29] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive query containment and answering under description logics constraints. *ACM Transactions on Computational Logic*, 2007. To Appear. [48, 49]

[30] D. Calvanese, T. Eiter, and M. Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proceedings of the 22th National Conference on Artificial Intelligence (AAAI 2007)*, 2007. [41, 44, 116, 139, 146, 147]

[31] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1994. [48]

[32] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing (STOC 1977)*, pages 77–90, New York, NY, USA, 1977. ACM Press and Addison Wesley. [48]

[33] S. de Coronado, M. W. Haber, N. Sioutos, M. S. Tuttle, and L. W. Wright. NCI thesaurus: Using science-based terminology to integrate cancer research results. In *Proceedings of MEDINFO 2004*. IOS Press, 2004. [16]

[34] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza", 1995. [63]

[35] G. De Giacomo and M. Lenzerini. Tbox and abox reasoning in expressive description logics. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the 5th International Conference on the Principles of Knowledge*

Representation and Reasoning (KR 1996), pages 316–327, San Francisco, California, 1996. Morgan Kaufmann, Los Altos. [49]

[36] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. *Principles of Knowledge Representation and Reasoning*, Studies in Logic, Language and Information, CSLI Publications:193–238, 1996. [14]

[37] A. Emmen. The grid needs ontologies—onto-what?, 2002. http://www.hoise.com/primeur/03/articles/monthly/AE-PR-02-03-7.html. [16]

[38] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 2st edition, 1996. [39, 61]

[39] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002. http://www.globus.org/research/papers/ogsa.pdf. [16]

[40] H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999)*, Washington, DC, USA, 1999. IEEE Computer Society Press. [67]

[41] B. Glimm. A query language for web ontologies, 2004. Bachelor Report, Hamburg University of Applied Sciences, http://www.cs.man.ac.uk/~glimmbx/download/report.pdf. [40, 169]

[42] B. Glimm and I. Horrocks. Handling cyclic conjunctive queries. In *Proceedings of the 2005 Description Logic Workshop (DL 2005)*, Edinburgh, Scotland, UK, 2005. CEUR (http://ceur-ws.org/). [27]

[43] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering in the description logic $\mathcal{SHIQ}$. LTCS-Report LTCS-06-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2006. http://www.cs.man.ac.uk/~glimmbx/download/GHLS06a.pdf. []

[44] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, 2006. [27, 28, 57]

[45] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering in the description logic $\mathcal{SHIQ}$. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007. [27]

[46] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query entailment for $\mathcal{SHOQ}$. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*, 2007. [28]

[47] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *Journal of Artificial Intelligence Research*, 31:151–198, 2008. [27]

[48] C. Goble, S. Pettifer, R. Stevens, and C. Greenhalgh. Knowledge Integration: *In Silico* Experiments in Bioinformatics. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 121–134. Morgan Kaufmann, Los Altos, 2003. [16]

[49] C. Golbreich and I. Horrocks. The OBO to OWL mapping, GO to OWL 1.1! In *Proceedings of the 3rd OWL Experiences and Directions Workshop (OWLED 2007)*, number 258 in CEUR Workshop Proceedings. CEUR (`http://ceur-ws.org/`), 2007. [16]

[50] C. Golbreich, S. Zhang, and O. Bodenreider. The foundational model of anatomy in OWL: Experience and perspectives. *John Wiley & Sons*, 4(3), 2006. [16]

[51] C. Golbreich, M. Horridge, I. Horrocks, B. Motik, and R. Shearer. OBO and OWL: Leveraging semantic web technologies for the life sciences. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2007)*, Lecture Notes in Computer Science. Springer-Verlag, 2007. [16]

[52] J. Goodwin. Experiences of using OWL at the ordnance survey. In *Proceedings of the 1st OWL Experiences and Directions Workshop (OWLED 2005)*, volume 188 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2005. [16]

[53] E. Grädel. Why are modal logics so robustly decidable? In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science, Entering the 21th Century*, volume 2, pages 393–408. World Scientific, 2001. [70]

[54] E. Grädel. Description logics and guarded fragments of first order logic. In *Proceedings of the 1998 Description Logic Workshop (DL 1998)*, CEUR Workshop Proceedings, 1998. [60]

[55] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999. [66]

[56] G. Grahne. *Problem of Incomplete Information in Relational Databases*. Springer-Verlag, 1991. [41, 46]

[57] V. Haarslev and R. Möller. Racer system description. In R. Gor, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 / 2001 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag, 2001. [11, 15]

[58] V. Haarslev, R. Möller, R. Van Der Straeten, and M. Wessel. Extended query facilities for Racer and an application to software-engineering problems. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 Description Logic Workshop (DL 2004)*, volume 14, pages 148–157, 2004. [16]

[59] I. Horrocks and P. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *John Wiley & Sons*, 1(4):345–357, 2004. [51]

[60] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In *Proceedings of the 13th International Conference on World Wide Web (WWW 2004)*, pages 723–731. ACM Press and Addison Wesley, 2004. [50]

[61] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(d) description logic. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001. [121, 133]

[62] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 2005. [34]

[63] I. Horrocks and U. Sattler. A tableau decision procedure for $\mathcal{SHOIQ}$. *Journal of Automated Reasoning*, 39(3):249–276, 2007. [60, 111, 121, 170]

[64] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 399–404, 2000. [44, 70, 84]

[65] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. Query containment using a DLR ABox. LTCS-Report LTCS-99-15, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1999. [25, 48, 49, 91]

[66] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR 1999)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999. [16, 31]

[67] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR 2000)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000. [48]

[68] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In D. McAllester, editor, *Proceedings of the 17th Conference on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000. [32, 34, 63, 71, 111]

[69] I. Horrocks, B. Glimm, and U. Sattler. Hybrid logics and ontology languages. *Electronic Notes in Theoretical Computer Science*, 174(6):3–14, 2007. Proceedings of the International Workshop on Hybrid Logic (HyLo 2006). [27]

[70] D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding semantic matching of stateless services. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1319–1324, 2006. [16]

[71] U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *Lecture Notes in Artificial Intelligence*, pages 191–205. Springer-Verlag, 2000. [85, 121]

[72] U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption testing with SPASS. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the 1999 Description Logic Workshop (DL 1999)*, pages 136–137. Linköping University, 1999. [11, 15, 85, 121]

[73] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proceedings of the 11th International Conference on Logic for Programming and Automated Reasoning (LPAR 2004)*, volume 3452 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004. [120]

[74] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162, Whistler, Canada, 2004. AAAI Press/The MIT Press. [66, 111, 141, 142]

[75] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005. [43]

[76] T. S. Kaczmarek, R. Bates, and G. Robins. Recent developments in NIKL. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI 1986)*, pages 978–985, 1986. [14]

[77] Y. Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006. [66]

[78] Y. Kazakov and B. Motik. A resolution-based decision procedure for $\mathcal{SHOIQ}$. In U. Furbach, J. Harrison, and N. Shankar, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130, pages 662–667, Seattle, WA, USA, 2006. Springer-Verlag. [66, 121, 141, 142, 170]

[79] Y. Kazakov, U. Sattler, and E. Zolin. How many legs do I have? Non-simple roles in number restrictions revisited. In *Proceedings of the 14th International Conference on Logic for Programming and Automated Reasoning (LPAR 2007)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007. [31]

[80] E. Kieronski. Results on the guarded fragment with equivalence or transitive relations. In *Computer Science Logic (CSL 2005), Proceedings of the 14th Annual Conference of the European Association for Computer Science Logic (EACSL)*, volume 3634 of *Lecture Notes in Computer Science*, pages 309–324. Springer-Verlag, 2005. [67]

[81] M. Kracht. How completeness and correspondence theory got married. In M. de Rijke, editor, *Diamonds and Defaults*, volume 229, pages 175–214. Kluwer Academic Publishers, 1993. [46]

[82] A. Krisnadhi and C. Lutz. Data complexity in the $\mathcal{EL}$ family of description logics. In N. Dershowitz and A. Voronkov, editors, *Proceedings of the 14th International Conference on Logic for Programming and Automated Reasoning (LPAR 2007)*, volume 4790 of *Lecture Notes in Artificial Intelligence*, pages 333–347. Springer-Verlag, 2007. [46]

[83] M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 310–323. Springer-Verlag, 2007. [46]

[84] L. Lacy, G. Aviles, K. Fraser, W. Gerber, A. Mulvehill, and R. Gaskill. Experiences using OWL in military applications. In *Proceedings of the 1st*

OWL Experiences and Directions Workshop (OWLED 2005), volume 188 of *CEUR Workshop Proceedings*. CEUR (`http://ceur-ws.org/`), 2005. [16]

[85] F. Lehmann. Semantic networks. *Semantic Networks in Artificial Intelligence*, pages 1–50, 1992. [14]

[86] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 1996)*, pages 323–327, 1996. [42, 52, 120]

[87] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998. [50, 51, 52, 165]

[88] C. Lutz. Inverse roles make conjunctive queries hard. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*, 2007. [47, 116, 163, 166]

[89] R. MacGregor and R. Bates. The LOOM knowledge representation language. Technical Report ISI/RS-87-188, The University of Southern California, Information Sciences Institute, 1987. [14]

[90] M. Minsky. A framework for representing knowledge. *Cognitive Science*, 1992. The original appeard as a longer version in 1974 as MIT-AI Laboratory Memo 306, reprinted in *The Psychology of Computer Vision, 1975* and a shorter version also appeared in *Mind Design, MIT Press, 1981.* [14]

[91] B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany, 2006. [66, 141, 142]

[92] B. Motik and U. Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In M. Hermann and A. Voronkov, editors, *Proceedings of the 13th International Conference on Logic for Programming and Automated Reasoning (LPAR 2006)*, volume 4246 of *Lecture Notes in Computer Science*, pages 227–241, Phnom Penh, Cambodia, 2006. Springer-Verlag. [11, 15]

[93] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, 2004. [45, 50, 52]

[94] B. Motik, P. F. Patel-Schneider, and I. Horrocks. OWL 1.1 web ontology language structural specification and functional-style syntax. URL, 2006. http://www.w3.org/Submission/owl11-owl_specification. [16]

[95] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987. [146]

[96] OBO. Open biomedical ontologies repository. http://www.bioontology.org/repositories.html#obo, 2007. [16]

[97] M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI 2006)*, 2006. [44, 45, 56, 58, 120, 165]

[98] M. M. Ortiz de la Fuente, D. Calvanese, T. Eiter, and E. Franconi. Data complexity of answering conjunctive queries over $\mathcal{SHIQ}$ knowledge bases, 2005. http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0507059. [11, 58, 165]

[99] L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997)*, Washington, DC, USA, 1997. IEEE Computer Society Press. [14]

[100] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM Journal of Computing*, 29(4):1083–1117, 2000. [66]

[101] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley Publ. Co., Reading, Massachussetts, 1994. [42]

[102] J. Quantz and K. Kindermann. Implementation of the BACK system version 4. KIT Report 78, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, 1990. [14]

[103] M. R. Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Readings in Knowledge Representations*, 12:410–430, 1967. Appears also in Behavioral Science 12:410-430. [14]

[104] A. Rector, W. Solomon, W. Nowlan, and T. Rush. A terminology server for medical language and medical information systems. *Methods of Information in Medicine*, 34:147–157, 1994. [16]

[105] R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005. [50]

[106] R. Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, 2006. [52, 119, 163, 167]

[107] R. Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In *Proceedings of the 25th ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS 2006)*, pages 356–365. ACM Press and Addison Wesley, 2006. [41, 46]

[108] R. Rosati. On conjunctive query answering in EL. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*. CEUR Workshop Proceedings, 2007. [46]

[109] R. Rosati. The limits of querying ontologies. In *Proceedings of the 11th International Conference on Database Theory (ICDT 2007)*, volume 4353 of *Lecture Notes in Computer Science*, pages 164–178. Springer-Verlag, 2007. [42, 45, 46]

[110] N. Rychtyckyj. DLMS: An evaluation of KL-ONE in the automobile industry. In *Proceedings of the 5th International Conference on the Principles of Knowledge Representation and Reasoning (KR 1996)*, pages 588–596, Cambridge, MA, 1996. Morgan Kaufmann, Los Altos. [16]

[111] H. Sahlqvist. Correspondence and completeness in the first- and second-order semantics for modal logic. In S. Kanger, editor, *Proceedings of the 3rd Scandinavian Logic Symposium*, pages 110–143, 1975. [46]

[112] S. Sahoo, O. Bodenreider, K. Zeng, and A. Sheth. An experiment in integrating large biomedical knowledge resources with rdf: Application to

associating genotype and phenotype information. In *Proceedings of International Workshop on Health Care and Life Sciences Data Integration for the Semantic Web*, 2007. [16]

[113] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, pages 76–91, London, UK, 2001. Springer-Verlag. [139, 146, 147]

[114] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278, 1993. [118]

[115] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994. [34, 120]

[116] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 466–471, Sidney, AU, 1991. [162, 163]

[117] R. A. Schmidt and U. Hustadt. A principle for incorporating axioms into the first-order translation of modal formulae. In *Proceedings of the 19th Conference on Automated Deduction (CADE 2003)*, volume 2741 of *Lecture Notes in Computer Science*, pages 412–426. Springer-Verlag, 2003. [111]

[118] R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 438–451. Springer-Verlag, 2007. [85, 121]

[119] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991. [12]

[120] S. Schulz and U. Hahn. Parts, locations, and holes - formal reasoning about anatomical structures. In *Proceedings of the 8th Conference on AI in Medicine in Europe (AIME 2001)*, volume 2101 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001. [16]

[121] A. Sidhu, T. Dillon, E. Chang, and B. S. Sidhu. Protein ontology development using OWL. In *Proceedings of the 1st OWL Experiences and*

Directions Workshop (OWLED 2005), volume 188 of *CEUR Workshop Proceedings*. CEUR (http://ceur-ws.org/), 2005. [16]

[122] E. Sirin and B. Parsia. Optimizations for answering conjunctive abox queries. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*, 2006. [40, 45, 169]

[123] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. Accepted for the Journal of Web Semantics, Available online at http://www.mindswap.org/papers/PelletJWS.pdf, 2006. [11, 15, 16]

[124] K. A. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue. [16]

[125] K. A. Spackman and K. E. Campbell. Compositional concept representation using SNOMED: Towards further convergence of clinical terminologies. In *Proceedings of AMIA Annual Fall Symposium*, pages 875–879, 1998. [16]

[126] K. A. Spackman, K. E. Campbell, and R. A. Côté. SNOMED RT: A reference terminology for health care. *Journal of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement. [16]

[127] SWEET. Semantic web for earth and environmental terminology (SWEET). Jet Propulsion Laboratory, California Institute of Technology, 2007. http://sweet.jpl.nasa.gov/. [16]

[128] W. Szwast and L. Tendera. On the decision problem for the guarded fragment with transitivity. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS 2001)*, Washington, DC, USA, 2001. IEEE Computer Society Press. [67]

[129] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001. [21, 22, 26, 38, 39, 40, 44, 62, 63, 81, 90, 120, 137]

[130] W. Thomas. *Languages, Automata, and Logic*. Springer-Verlag, New York, NY, USA, 1997. [146]

[131] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000. [14]

[132] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001. [14, 43, 47, 66, 105, 109, 110, 111, 113, 114, 118, 121, 146, 163, 166, 169]

[133] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In U. Furbach and N. Shankar, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Computer Science*, pages 292 – 297. Springer-Verlag, 2006. [11, 15, 16]

[134] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open grid services infrastructure (ogsi) version 1.0, 2003. `http://www.ogf.org/documents/GFD.15.pdf`. [16]

[135] J. van Benthem. Dynamic bits and pieces. Technical Report LP-1997-01, Universiteit van Amsterdam, Institute for Logic, Language and Computation, 1997. [67]

[136] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. In *Proceedings of the 11th ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS 1992)*, pages 331–345. ACM Press and Addison Wesley, 1992. [40]

[137] R. van der Meyden. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publishers, 1998. [41, 46]

[138] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC 1982)*, pages 137–146. ACM Press and Addison Wesley, 1982. [42]

[139] M. Y. Vardi. Alternating automata and program verification. In J. van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 471–485. Springer-Verlag, 1995. [147, 161]

[140] M. Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop*, volume 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1997. [23, 70]

[141] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, pages 628–641, London, UK, 1998. Springer-Verlag. [146, 147]

[142] C. Weidenbach, R. Schmidt, T. Hillenbrand, R. Rusev, and D. Topic. Spass version 3.0. In *Proceedings of the 21th Conference on Automated Deduction (CADE 2007)*, volume 4603 of *Lecture Notes in Computer Science*, pages 514–520, Bremen, Germany, 2007. Springer-Verlag. [11, 15, 85, 121]

[143] M. Wessel and R. Möller. A high performance semantic web query answering engine. In *Proceedings of the 2005 Description Logic Workshop (DL 2005)*, 2005. [45, 169]

[144] K. Wolstencroft, A. Brass, I. Horrocks, P. Lord, U. Sattler, D. Turi, and R. Stevens. A Little Semantic Web Goes a Long Way in Biology. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2005)*, 2005. [16]

[145] K. Wolstencroft, R. McEntire, R. Stevens, L. Tabernero, and A. Brass. Constructing Ontology-Driven Protein Family Databases. *Bioinformatics*, 21(8):1685–1692, 2005. [16]

[146] W. A. Woods. What's in a link: Foundations for semantic networks. *Representation and Understanding: Studies in Cognitive Science*, pages 35–82, 1975. [14]

[147] C. Wroe, J. J. Cimino, and A. L. Rector. Integrating existing drug formulation terminologies into an HL7 standard classification using OpenGALEN.

Journal of the American Medical Informatics Association, 2001. Special Conference Issue. [16]

[148] C. Wroe, C. A. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating experiments using semantic data on a bioinformatics grid. *IEEE Intelligent Systems*, 19(1), 2004. Special Issue on e-Science. [16]

[149] E. Zolin. Query answering based on modal correspondence theory. In *Proceedings of the 4th Methods for Modalities Workshop (M4M4)*, pages 21–37, 2005. [46, 47]

[150] E. Zolin. Modal logic applied to query answering and the case for variable modalities. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*. CEUR Workshop Proceedings, 2007. [46, 47]

Index