Technische Universität Dresden
International Master Programme in Computational Logic
Institute for Theoretical Computer Science
Department of Computer Science

Master Thesis

Data Complexity of Instance Checking in The $\mathcal{EL}$ Family of Description Logics

Adila Alfa Krisnadhi
Born on September 6, 1980 in Pringsewu, Indonesia

Submission date: March 9, 2007

Overseeing Professor: Prof. Dr. Franz Baader
Supervisor: Dr. Carsten Lutz

TECHNISCHE UNIVERSITÄT DRESDEN

Author	:	**Adila Alfa Krisnadhi**
Matrikel-Nr.	:	**3177963**
Title	:	**Data Complexity of Instance Checking in The $\mathcal{EL}$ Family of Description Logics**
Degree	:	**Master of Science**
Date of submission	:	**March 9, 2007**

Declaration

Hereby I certify that the thesis has been written by me. Any help that I have received in my research work has been acknowledged. Additionally, I certify that I have not used any auxiliary sources and literature except those cited in the thesis.

Adila Alfa Krisnadhi

Abstract

Subsumption in the description logic (DL) $\mathcal{EL}$ is known to be tractable even when it is done with respect to the most general form of terminology, namely a set of general inclusion axioms (GCIs). Recently, this tractability boundary has been clarified by identifying DL constructors that causes intractability of subsumption when added to $\mathcal{EL}$ and that do not. These results provide us with a characterization of the complexity of subsumption for the $\mathcal{EL}$ family of DLs (i.e., $\mathcal{EL}$ and its extensions).

Besides subsumption, there are other standard reasoning problems studied in DL. Among them, the instance checking problem is the most basic reasoning problem that is concerned with deriving implicit knowledge about individuals in a DL knowledge base. Such a knowledge base consists of an intensional part in the form of a terminology (TBox) and an extensional or data part in the form of assertions about particular individuals in the domain of the knowledge base (ABox). Like other reasoning problems, complexity of instance checking is usually measured in the size of the whole input—thus called combined complexity—which, in this case, consists of a TBox, an ABox, a query concept and an individual name. On the other hand, it is common to assume that the data (ABox) is very large compared to the TBox and the query. Therefore, it is often more realistic to use a complexity measure based only on the size of the ABox, i.e., data complexity.

For the $\mathcal{EL}$ family, results for the combined complexity of instance checking can be derived from the complexity results for subsumption. But results which are concerned with data complexity are still lacking. This motivates us to investigate the data complexity of instance checking in the $\mathcal{EL}$ family. In particular, we are interested in whether there are extensions of $\mathcal{EL}$ which are intractable regarding combined complexity, but tractable regarding data complexity.

The first part of this thesis establishes coNP-hardness (and even coNP-completeness) results regarding data complexity of instance checking w.r.t. sets of GCIs for extensions of $\mathcal{EL}$ with negation, disjunction, value restriction, number restriction and role constructors such as role negation, role union and transitive closures. The lower bounds of data complexity for these DLs are proved by polynomial reductions from the complement of 2+2-SAT, a variant of propositional satisfiability problem which is NP-complete, whereas the upper bounds follow from known results of data complexity for $\mathcal{ALC}$ and $\mathcal{SHIQ}$.

The second part identifies an extension of $\mathcal{EL}$ called $\mathcal{ELI}^f$, for which data complexity of instance checking w.r.t. sets of GCIs is tractable. The DL $\mathcal{ELI}^f$ is obtained from $\mathcal{EL}$ by adding inverse roles and global functionality. This result is interesting since adding only one of those two constructors leads to intractability of reasoning w.r.t. combined complexity. The result is derived by giving an algorithm that decides instance checking in $\mathcal{ELI}^f$ w.r.t. sets of GCIs and runs in time polynomial in the size of the input ABox.

Acknowledgements

Alhamdulillah! I would not have been able to write my thesis without You, Allah the One Lord of the Whole Universe, thus my first and foremost gratitude eternally goes to Him. I thank Him for the guidance, the love, the life and the belief in You of which I pray to last the whole of my lifetime.

I am greatly thankful to my family. My wife, Tuntas, for her prayers, enthusiasm, love, support, and even for just being there for me. My parents, who pray from far away, days and nights for my success. My other relatives whose names would be a very long list to be written here, I thank them all for their support and suggestions virtually whenever I need.

Many many thanks goes to Carsten Lutz with his patience and enthusiasm of helping and guiding me, not only to finally finish this thesis but also from the time when I was working on my CL-project. Not only once or twice, but close to infinitely many times has he given me stimulating advices and questions showing me the right way to finish my work. Additionally, I am grateful for his willingness to proofread my thesis and to endure plentiful of mistakes and imprecisions in it, and still to come up with numerous useful criticisms.

I would like to thank Prof. Franz Baader for his interesting lectures on Logic-Based Knowledge Representation, and Complexity and Logic. With the first, I was introduced for the first time to the world of Description Logics and with the second, I learned a lot of things about theoretical computer science, in particular complexity theory and its relation to logics.

In addition, I appreciate and feel honored by the opportunity which has been given by German Academic Exchange Service (DAAD) and Faculty of Computer Science, University of Indonesia through the scholarship, enabling me to complete the Master's degree in Germany. I am also greatly indebted to the International Master Programme in Computational Logic, TU Dresden for the eventful time of study, and also for numerous new knowledge and experience I acquired during my stay in Dresden. Special thanks also goes to the following professors: Steffen Hölldobler and Horst Reichel of TU Dresden, also L. Yohanes Stefanus and Benyamin Kusumoputro of University of Indonesia, for their "gutachten" with which my more-than-two years stay in Dresden was possible.

Contents

Chapter 1

Introduction

Knowledge Representation (KR) is one of the most prominent subfield in Artificial Intelligence (AI). Research in KR is mainly focused on building systems that possess the ability to find implicit consequences from explicitly represented knowledge. Such systems are known as knowledge-based systems [Nardi & Brachman, 2003].

One of the earliest approaches was the use of *network-based structures* [Lehmann, 1992], such as *semantic networks* [Quillian, 1968] and *frames* [Minsky, 1974]. This approach aimed at representing sets of individuals and their relationship by means of the structure of the network. The intuition was that, with such structures, representation could be done in a simple way, and reasoning would be efficient.

Both semantic networks and frames were based on the use of graphical interfaces in which knowledge is represented with the help of ad-hoc data structures and the reasoning is accomplished by similarly ad-hoc procedures which manipulate the data structures. Initial attempts on realizing these formalisms suffered from the lack of precise semantic characterization. Due to this weakness, one cannot guarantee the same behavior of two KR systems which are built from identical-looking components and identical relationship names. This gave rise to the line of research on providing a semantics for such representation structures in such a way that ease of representation and efficiency of reasoning could still be retained.

The first important step in this direction was the recognition that frames could be given semantics relying on first-order logic [Hayes, 1979]. Further investigations showed that frames and semantics networks could indeed be regarded as fragments of first-order logic and different features of the representation language would lead to different fragments of first-order logic [Brachman & Levesque, 1985]. Consequently, one could develop specialized reasoning techniques without necessarily requiring the power of general first-order logic theorem provers. Furthermore, reasoning in different fragments of first-order logic leads to different problems in computational complexity. This became the origin of the research on description logics that was originated from the work on the KL-ONE system [Brachman & Schmolze, 1985].

1.1 Description Logics

Description logics (DLs) [Baader *et al.*, 2003] are a family of logic-based knowledge representation formalisms designed to represent and reason about conceptual knowledge. As a family of logical formalisms, DLs are equipped with a well-defined—usually Tarski style, extensional—semantics and provide a logical basis for interpreting objects (or *individuals*), classes of objects (or *concepts*), and relationships between objects (or *roles*). A particular DL is typically characterized by the set of *constructors* with which complex *concept descriptions* can be built from atomic *concept names* and *role names.* For example, $\mathcal{ALC}$ (Attributive Language with Complement) [Schmidt-Schauß & Smolka, 1991] is the smallest propositionally closed Description Logic. It provides all Boolean operators, value restriction, i.e., universal restriction, and existential restriction. The following is an $\mathcal{ALC}$-concept which describes fathers who has at least two children: a son and a daughter, and all the children are rich, using concept conjunction ($\sqcap$), negation ($\neg$), and value restriction ($\forall$) and existential restriction ($\exists$) over the role $\mathsf{has\_child}$.

$$\mathsf{Male} \sqcap \exists \mathsf{has\_child}.\mathsf{Male} \sqcap \exists \mathsf{has\_child}.\neg \mathsf{Male} \sqcap \forall \mathsf{has\_child}.\mathsf{Rich}$$

Semantically, concepts are given a set-theoretic interpretation. More precisely, concepts are interpreted as sets of individuals and roles are interpreted as binary relations. Each DL constructor denotes a particular set construction giving a set of individuals from sets of individuals denoted by concepts and roles on which the constructor is applied. For example, concept conjunction is interpreted as set intersection and negation is interpreted as set complement. [Nardi & Brachman, 2003]. All individuals come from the domain of interpretation which can be chosen arbitrarily and can be infinite. The non-finiteness of the domain and the *open-world-assumption* are important features of DLs in comparison to modeling languages studied in databases [Baader & Nutt, 2003; Borgida *et al.*, 2003; Sattler *et al.*, 2003].

Description Logics Knowledge Base: TBox and ABox

The knowledge base of a DL system is made up of two components, namely a *terminological component* or *TBox* and an *assertional component* or *ABox* [Baader & Nutt, 2003]. A TBox is given in the form of a terminology (hence the term "TBox") which contains intensional knowledge (or general knowledge) of the problem and describes properties of concepts that hold in general over the domain. In contrast, the ABox contains extensional knowledge or assertional knowledge (hence the term "ABox") which is basically knowledge that is specific to individuals in the domain of the problem.

A TBox is a set of declarative statements. These statements come in two forms, namely *concept definitions* and *(general) concept inclusions* (*GCIs*). A concept definition is a statement of the form $A \equiv C$ which defines a *concept name* A in terms of a possibly complex concept C. Such a definition is interpreted as *logical equivalence* and that gives sufficient and necessary conditions for classifying an individual as a member of the set denoted by A. A GCI is a statement of the form $C \sqsubseteq D$ over arbitrary concepts C and D, which is interpreted as a *logical implication.* Notably, a concept definition $A \equiv C$ can be expressed using two GCIs: $A \sqsubseteq C$ and $C \sqsubseteq A$. Due to this, one can view every TBox as simply a set of GCIs.

In the literature [Baader & Nutt, 2003], one distinguishes different kinds of TBoxes depending on some syntactic restrictions. The simplest, i.e., the most restrictive ones are the so-called *acyclic* TBoxes. An acyclic TBox contains only concept definitions such that for every concept name, no more than one definition is allowed and additionally, the TBox itself has no terminological cycle (thus the term "acyclic"). A terminological cycle occurs whenever a concept name is defined directly or indirectly in terms of itself. If we drop the acyclicity restriction, then the TBoxes are called *cyclic* TBoxes. Finally, the most general form of TBoxes are the so-called *general* TBoxes which may contain GCIs. As an example, consider the following TBox that formalizes some knowledge about relationships between people, where $\bot$ is interpreted as the empty set. Besides constructors mentioned in the previous example, the following example also uses disjunction ($\sqcup$) which is interpreted as set union.

$$\begin{aligned}
\mathsf{Human} &\equiv \mathsf{Male} \sqcup \mathsf{Female}\\
\mathsf{Parent} &\equiv \mathsf{Human} \sqcap \exists \mathsf{has\_child}.\mathsf{Human} \sqcap \forall \mathsf{has\_child}.\mathsf{Human}\\
\mathsf{Father} &\equiv \mathsf{Parent} \sqcap \mathsf{Male}\\
\mathsf{Mother} &\equiv \mathsf{Parent} \sqcap \mathsf{Female}\\
\mathsf{Husband} &\equiv \mathsf{Male} \sqcap \exists \mathsf{married\_to}.\mathsf{Human} \sqcap \forall \mathsf{married\_to}.\mathsf{Female}\\
\mathsf{Wife} &\equiv \mathsf{Female} \sqcap \exists \mathsf{married\_to}.\mathsf{Human} \sqcap \forall \mathsf{married\_to}.\mathsf{Male}\\
\mathsf{MarriedHuman} &\equiv \mathsf{Husband} \sqcup \mathsf{Wife}\\
\mathsf{Male} \sqcap \mathsf{Female} &\sqsubseteq \bot
\end{aligned}$$

The first seven statements in this example introduce Human, Parent, Father, Mother, Husband, Wife and MarriedHuman in terms of of other concepts. The last statement introduces the disjointness of concepts Male and Female.

An ABox is a set of assertions which formalizes (parts of) a specific situation involving certain individuals. Every assertion asserts a fact about one or more individuals denoted by individual names. There are two forms of assertions which can be expressed in an ABox, namely *concept assertions* and *role assertions*. A concept assertion is an assertion of the form $C(a)$ stating the fact that the individual denoted by the individual name a belongs to the set denoted by the concept C. A role assertion is a statement of the form $r(a, b)$ asserting the fact that the individual a is related to the individual b via the role r. The following is an example of an ABox which gives a partial description of a particular family. In this example, ANDRE, STEFFI and JAZ are individual names.

$$\begin{array}{lll}
(\mathsf{Male} \sqcap \mathsf{Parent})(\mathsf{ANDRE}), & \mathsf{Wife}(\mathsf{STEFFI}), & \mathsf{married\_to}(\mathsf{STEFFI}, \mathsf{ANDRE}),\\
\mathsf{has\_child}(\mathsf{STEFFI}, \mathsf{JAZ}), & \mathsf{has\_child}(\mathsf{ANDRE}, \mathsf{JAZ}) &
\end{array}$$

Reasoning in Description Logics

As a KR formalism, DLs are useful not merely for representing knowledge of a particular domain, but also for reasoning about that represented knowledge, i.e., deriving implicit knowledge from the knowledge that is explicitly stated in the knowledge base. In the literature [Baader & Nutt, 2003], there are several reasoning tasks for DLs that have been considered. Among others, the following ones are considered as basic reasoning tasks: *concept satisfiability*, *subsumption*, *knowledge base consistency (satisfiability)*, and *instance checking*. All these problems are casted as decision problems and take a knowledge base as their input. In addition to the knowledge base, concept satisfiability and subsumption respectively take one and two input concepts, whereas instance checking take an input concept and an individual. Concept satisfiability decides whether there is an interpretation satisfying the knowledge base which interprets the given concept as a nonempty set. Subsumption decides whether for every interpretation that satisfies the knowledge base, every instance of the first concept is also an instance of the second concept. Knowledge base consistency simply determines if there is an interpretation satisfying the knowledge base. Finally, instance checking decides whether for every interpretation satisfying the knowledge base, the input individual is an instance of the input concept.

Out of those four basic reasoning tasks, concept satisfiability and subsumption are the reasoning tasks for which ABoxes have no effect [Buchheit *et al.*, 1993; Nebel, 1990a]. For the other two problems, ABoxes does affect reasoning. Knowledge base consistency typically concerns with the interaction between TBox and ABox in the knowledge base, whereas instance checking is the basic reasoning task for deriving implicit knowledge about individuals in the knowledge base [Schaerf, 1993]. It is well-known that concept satisfiability, subsumption and knowledge base consistency can be reduced to instance checking [Donini *et al.*, 1994]. In addition, there is another reasoning task that can be viewed as a generalization to instance checking, namely *query answering* or *retrieval*. Query answering is the reasoning problem that can be stated as follows: "given a concept and a knowledge base, find all instances of the given concept in the knowledge base". It is easy to see that query answering can be reduced as a set of instance checking tests and conversely, instance checking can be seen as a special case of query answering involving only one individual. Due to all these reasons, we focus on instance checking as the main theme for this thesis.

Complexity Analysis in Description Logics

When analyzing a reasoning problem in DL from a theoretical point of view, one is interested in finding the exact worst-case complexity of the problem and devising an algorithm for solving the problem such that its computational complexity match the worst-case complexity of the problem [Tobies, 2001]. The standard notion of complexity which is usually considered in the literature is measured in the size of the whole input of the reasoning problem of interest. For many cases, this standard notion is sufficient for characterizing the complexity of the problem and providing a good measure of the performance of the algorithm for solving the reasoning problem in the context.

However, for instance checking whose input consists of a TBox, an ABox, a *query* concept, and an individual name, one can also consider a complexity measure that is based *only* the size of the ABox. For applications in which the size of the ABox is much larger than the size of the TBox and the query concept, this complexity notion can give a more realistic measure of the difficulty of the problem of instance checking, and the performance of algorithms for solving it [Calvanese *et al.*, 2006; Hustadt *et al.*, 2005].

This above distinction between different complexity notions for instance checking is analogous to the way complexity of query answering is characterized in the database theory. In fact, there is a clear correspondence between querying a relational database and instance checking as a basic form of querying a DL knowledge base. In the problem of querying a relational database, one considers the input of the problem to consist of one part that constitutes the *data* and another part that constitutes a *query* expression. Moreover, in such situation, the data must obey a kind of structure in the form of database *schema*. In the context of instance checking in DLs, the ABox and the TBox can be seen respectively as the data and the schema part, whereas the query concept corresponds to the query expression in database. Note that though, in the context of database, the schema is always considered fixed for any situation of database query, whereas in the context of DLs, the TBox may vary and thus, unlike database schema, is considered as part of input of the problem.

For complexity analysis in querying a relational database, one distinguishes between *combined complexity* which is measured in the size of the whole input, i.e., the data and the query expression, *data complexity* which is measured only in the size of the data and *query complexity* which is measured only in the size of the query [Vardi, 1986][1]. There, data complexity is used as a more realistic performance measure for cases when the size of data is much larger than the size of the query. Likewise, query complexity is a more realistic performance measure for cases in which the size of query expression is much larger then the size of the data.

By taking into account the correspondence between DL knowledge base and database, those complexity notions from database theory described above are borrowed to characterize the complexity of instance checking. Thus, we use the term *combined complexity* for the standard notion of complexity of reasoning in DL that is usually considered in the literature, *data complexity* for the complexity measure that is based only on the size of the ABox, and *query complexity* for the complexity measure that is based only on the size of the query concept. Note that since the schema is always fixed in database, no complexity measure is based only on the size of schema. Hence, no particular term is used for complexity measure that is based only the size of the TBox.

For application of DLs, the analogy to database theory described above is interesting because the idea of using ontologies (knowledge bases), as a conceptual view over data repositories is becoming more popular recently. For example, this is used in Enterprise Application Integration Systems, Data Integration Systems [Lenzerini, 2002] and The Semantic Web [Heflin & Hendler, 2001], where the data consists of instances of ontologies and its size is typically very large compared to the size of the intentional level of ontologies. For this reason, the notion of data complexity is also important to DL research and considered in this thesis.

Finally, note that, despite a close correspondence between DL knowledge bases and relational databases explained above, there is an important difference between them. In a database, the *closed world assumption* is typically assumed, thus admitting only a single model [Reiter, 1984]. This is in contrast to DL knowledge bases. A DL knowledge

[1]In [Vardi, 1986], query complexity is actually called *expression complexity*

base typically assumes the *open world assumption* as first-order theory, thus admitting an arbitrary (possibly infinite) number of models. This is the reason why reasoning in DL is intractable in many cases, whereas querying a relational (physical) database is polynomial w.r.t. data complexity [Schaerf, 1994].

In addition, we also do not consider the notion of query complexity for analysis in this thesis. This is due to the fact that cases in which the size of knowledge base is negligible compared to the size of the query concept are uncommon in applications.

1.2 The $\mathcal{EL}$ Family and Data Complexity: Previous Results

For historical reasons, DLs with (qualified) existential restriction ($\exists r.C$) but not value restriction ($\forall r.C$) were not until recently explored [Baader *et al.*, 2005a]. The reason was that in early DLs which were rooted in semantics networks and frames, it was decided that arcs in semantic networks and slots in frames should be perceived as value restrictions rather than existential restrictions. This was also the reason why the search for tractable DLs (DLs for which reasoning is polynomial-time decidable w.r.t. combined complexity) which was started after the first intractability results were shown in the 1980s [Brachman & Levesque, 1984; Nebel, 1988] was focused on extending the basic language $\mathcal{FL}_0$ which allows for the top-concept, conjunction and value restriction. For the subsumption problem without terminologies, the tractability barrier was investigated in detail in the early 1990s [Donini *et al.*, 1991]. However, further investigations showed that although $\mathcal{FL}_0$ is tractable without TBoxes [Brachman & Levesque, 1984], it is intractable when terminologies are involved, as it is coNP-complete for acyclic TBoxes [Nebel, 1990b], PSpace-complete for cyclic TBoxes [Baader, 1996] and ExpTime-complete for general TBoxes [Baader *et al.*, 2005a].

On the other hand, the DL $\mathcal{EL}$ which allows for the top-concept ($\top$), conjunction ($C \sqcap D$) and existential restriction ($\exists r.C$) is a simple DL for which subsumption is tractable w.r.t. acyclic and cyclic TBoxes [Baader, 2003b] and even w.r.t. general TBoxes [Brandt, 2004b]. This tractability of $\mathcal{EL}$ also holds for instance checking w.r.t. acyclic and cyclic TBoxes [Baader, 2003a] and w.r.t. general TBoxes [Brandt, 2004a].

These results motivated investigations of the $\mathcal{EL}$ *family*, i.e., DLs which are obtained by extending $\mathcal{EL}$ using various DL constructors. Recent results in [Baader *et al.*, 2005a,b] presented the "largest" extension of $\mathcal{EL}$ with standard DL constructors for which the subsumption problem can be decided in polynomial time even in the presence of GCIs. This DL, named $\mathcal{EL}^{++}$, is defined by extending $\mathcal{EL}$ with the bottom-concept, nominals and concrete domains. Furthermore, an $\mathcal{EL}^{++}$ terminology allows not only GCIs but also statements called role inclusion. In addition, Baader *et al.* [2005a,b] also identified some intractable extensions of $\mathcal{EL}$ which include extensions with negation, disjunction, value restriction, number restrictions, functionality, inverse roles, role negation, role union, and transitive closures. For all of the aforementioned extensions of $\mathcal{EL}$, subsumption w.r.t. general TBoxes were proved to be ExpTime-complete.

The choice of investigating $\mathcal{EL}$ and its extensions is also based on the fact that there are applications where the expressive power of $\mathcal{EL}$ or small extensions thereof appear to be sufficient. For example, the Systematized Nomenclature of Medicine (Snomed) [Cote *et al.*, 1993] corresponds to an acyclic $\mathcal{EL}$ TBox [Spackman, 2001]. Large part of the medical knowledge base Galen is actually expressible in $\mathcal{EL}$ with GCIs and transitive roles [Rector & Horrocks, 1997]. Finally, the Gene Ontology [The Gene Ontology Consortium, 2000] can actually be viewed as an acyclic $\mathcal{EL}$ TBox with one transitive role.

Concerning data complexity, there are some previous results that are worth mentioning here. One of the earliest quite comprehensive treatment of data complexity in DLs was given in the PhD thesis from Schaerf [1994]. In Schaerf's thesis, combined complexity and data complexity were analyzed for extensions of the DL $\mathcal{FL}^-$, the DL that provides conjunction, value restriction, and *unqualified* existential restrictions ($\exists r.\top$). These include the DL $\mathcal{AL}$ which is obtained from $\mathcal{FL}^-$ by adding the top-concept, the bottom concept and negation of concept names (*atomic* negation), as well as extensions of $\mathcal{AL}$ with qualified existential restrictions, disjunction, (full) negation, number restrictions, nominals, role conjunction, role chain, inverse roles, and role filler. Two more recent papers about data complexity in DLs are due to [Hustadt *et al.*, 2005] and [Calvanese *et al.*, 2006]. Hustadt *et al.* [2005]'s paper dealt with data complexity of reasoning in the DL $\mathcal{SHIQ}$, an extension of the DL $\mathcal{ALC}$ with transitive roles, role hierarchy, inverse roles and qualified number restrictions, whereas the paper from Calvanese *et al.* [2006] studied the data complexity of conjunctive query answering for the family of *DL-Lite* languages and its polynomial tractability boundaries.

1.3 Objective and Structure of the Thesis

This thesis aims to map out the data complexity of instance checking in the $\mathcal{EL}$ family of DLs. We basically present two kinds of results. First, we derive several intractability results regarding the data complexity of instance checking w.r.t. general TBoxes in the $\mathcal{EL}$ family. More precisely, we identify a number of extensions of $\mathcal{EL}$ for which instance checking w.r.t. general TBoxes may be harder than polynomial regarding data complexity. Second, we provide a tractability result on the data complexity of instance checking w.r.t. general TBoxes in the $\mathcal{EL}$ family. We identify an extension of $\mathcal{EL}$ for which data complexity of instance checking w.r.t. general TBoxes is polynomial. In order to prove this result, we present an algorithm deciding instance checking that runs in time polynomial in the size of the input ABox. The subsequent chapters are thus organized as follows.

In Chapter 2, we introduce the syntax and semantics of $\mathcal{EL}$ along with additional constructors which are used to define several extensions of $\mathcal{EL}$. Next, we introduce DL knowledge bases together with their standard semantics. Thereafter, we introduce the reasoning tasks and notions of complexity measures, especially instance checking and the notion of data complexity.

In Chapter 3, we present several intractability results for the $\mathcal{EL}$ family with respect to data complexity. More precisely, we show coNP-hardness (and even coNP-completeness) regarding data complexity of instance checking w.r.t. general TBoxes in several extensions of $\mathcal{EL}$ which include extensions with negation, disjunction, value restriction, number restrictions, and some role constructors: role complement, role union and transitive closures. The method to derive coNP-hardness is adapted from the method used by [Schaerf, 1993] to derive coNP-hardness of data complexity in $\mathcal{ALE}$. Whereas, for the coNP upper bound, we use the result from [Hustadt *et al.*, 2005] which established coNP-completeness regarding data complexity of instance checking for DLs providing at least the constructors from the DL $\mathcal{ALC}$ and at most the constructors from the DL $\mathcal{SHIQ}$.

Chapter 4 is dedicated to $\mathcal{ELI}^f$, the extension of $\mathcal{EL}$ with inverse roles and global functionality. We show that in this DL, data complexity of instance checking w.r.t. general TBoxes is polynomial. To obtain this result, a sound and complete tableau algorithm that decides instance checking in $\mathcal{ELI}^f$ and runs in time polynomial in the size of the input ABox is presented. This result is interesting because adding either inverse roles or global functionality yields to an ExpTime-completeness of subsumption w.r.t. general TBoxes [Baader *et al.*, 2005a,b].

Finally, we summarize this work in Chapter 5. There, we also discuss possible further work related to the main subject of this thesis.

Chapter 2

General Framework

In this chapter, we present the $\mathcal{EL}$-family of description logics. We start with introducing the syntax and semantics of the DL $\mathcal{EL}$ and its extensions. Then, we introduce DL knowledge bases. Finally, we introduce inference problems and complexity measures which are relevant for this work. In particular, we focus on the instance checking problem and the notion of data complexity.

2.1 The $\mathcal{EL}$ family

We introduce the syntax and semantics of the DL $\mathcal{EL}$ and subsequently, present various constructors which can be used to extend $\mathcal{EL}$ into other members of the $\mathcal{EL}$ family.

Definition 2.1 (Syntax of $\mathcal{EL}$-concepts)
Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be disjoint sets of *concept names* and *role names*. The set of *$\mathcal{EL}$-concept descriptions* (or *$\mathcal{EL}$-concepts*) is the smallest set that is inductively defined as follows:

- each $A \in \mathsf{N_C}$ is an (atomic) $\mathcal{EL}$-concept;
- if C, D are $\mathcal{EL}$-concepts and $r \in \mathsf{N_R}$ is a role name, then the top-concept $\top$, the conjunction $C \sqcap D$ and the existential restriction $\exists r.C$ are also $\mathcal{EL}$-concepts. $\diamond$

Definition 2.2 (Semantics of $\mathcal{EL}$-concepts)
An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$. The domain $\Delta^{\mathcal{I}}$ is simply a nonempty set and its elements are called *individuals.* The interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For a role r, we say that y is an *r-filler* of x whenever $(x, y) \in r^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to nonatomic concepts as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &\coloneqq \Delta^{\mathcal{I}}\\
(C \sqcap D)^{\mathcal{I}} &\coloneqq C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
(\exists r.C)^{\mathcal{I}} &\coloneqq \{x \in \Delta^{\mathcal{I}} \mid \exists y\colon (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}
\end{aligned}$$

A concept C is *satisfiable* iff there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. In this case, we say that $\mathcal{I}$ is a *model* of C. A concept C is *subsumed* by a concept D (written $C \sqsubseteq D$) iff for every interpretation $\mathcal{I}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Two concepts C, D are *equivalent* (written $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$. $\diamond$

Name	**Syntax**	**Semantics**
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} \colon (x,y) \in r^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$
at-least restriction	$(\geq n\ r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$
at-most restriction	$(\leq n\ r)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$

Table 2.1: Syntax and semantics of various concept constructors

Name	**Syntax**	**Semantics**
inverse roles	r^-	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\}$
role complement	$\neg r$	$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus r^{\mathcal{I}}$
role union	$r \cup s$	$r^{\mathcal{I}} \cup s^{\mathcal{I}}$
transitive closure	r^+	$\bigcup_{n\geq 1} (r^{\mathcal{I}})^n$

Table 2.2: Syntax and semantics of various role constructors

Several extensions of $\mathcal{EL}$ can be obtained by adding various constructors given in Table 2.1 and Table 2.2 to $\mathcal{EL}$. In these tables, C, D denote possibly complex concepts, r, s are possibly complex roles, n is a nonnegative integer and x, y are individuals. For a set S, $\#S$ denotes the cardinality of S.

For example, let Human, Female and Rich be concept names, and has_child a role name. The first concept below is expressed in $\mathcal{EL}$ and describes the notion of "parent". The other concepts are expressed in extensions of $\mathcal{EL}$ with various constructors presented in Table 2.1 and Table 2.2. They respectively describe the notion of "parent of a son", "parent who has a child that is either female or rich", "parent who has only rich children", "parent who has precisely two children", "person who has a rich parent", and "person whose one of his decendants (children, grandchildren, etc.) is rich".

$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}.\mathsf{Human}$$
$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}.(\mathsf{Human} \sqcap \neg \mathsf{Female})$$
$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}.(\mathsf{Female} \sqcup \mathsf{Rich})$$
$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}.\mathsf{Human} \sqcap \forall \mathsf{has\_child}.\mathsf{Rich}$$
$$\mathsf{Human} \sqcap (\geq 2\ \mathsf{has\_child}) \sqcap (\leq 2\ \mathsf{has\_child})$$
$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}^-.\mathsf{Rich}$$
$$\mathsf{Human} \sqcap \exists \mathsf{has\_child}^+.\mathsf{Rich}$$

2.2 Description Logic Knowledge Base: TBox and ABox

We first introduce the intensional component of a description logic knowledge base, called the TBox. A TBox describes how concepts are related to each other.

Definition 2.3 (TBox)
A *terminological axiom* is an expression of the form $A \equiv C$ (called *concept definition*) or $C \sqsubseteq D$ (called *general concept inclusion (GCI)*), where A is a concept name and C, D are concepts. A *(general) TBox* is a finite set of terminological axioms. Let $\mathcal{T}$ be a TBox that contains only concept definitions. We say that $\mathcal{T}$ contains *multiple definitions* iff there are two distinct concepts C_1 and C_2 such that both $A \equiv C_1$ and $A \equiv C_2$ belong to $\mathcal{T}$. We also say that $\mathcal{T}$ contains a *terminological cycle* iff there is a subset $\{A_1 \equiv C_1, \ldots, A_n \equiv C_n\} \subseteq \mathcal{T}$ such that

- A_{i+1} appears in C_i, for $1 \leq i \leq n$, and
- A_1 appears in C_n.

$\mathcal{T}$ is called an *acyclic TBox* iff it contains no multiple definition and no terminological cycle.

An interpretation $\mathcal{I}$ *satisfies* a concept definition $A \equiv C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$ and a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It *satisfies* a TBox $\mathcal{T}$ iff it satisfies every concept definition and GCI in $\mathcal{T}$. In this case, $\mathcal{T}$ is *satisfiable* and we say that $\mathcal{I}$ is a *model* of $\mathcal{T}$.

A concept C is *satisfiable w.r.t. a TBox* $\mathcal{T}$ iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept C is *subsumed* by a concept D *w.r.t. a TBox* $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Two concepts C, D are *equivalent w.r.t. a TBox* $\mathcal{T}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. ◇

Note that, due to the semantics, every concept definition $A \equiv C$ can be expressed by two GCIs: $A \sqsubseteq C$ and $C \sqsubseteq A$. Next, we introduce the ABox, the extensional component of a description logic knowledge base. An ABox describes a specific state of an application domain in terms of concepts and roles. It makes particular individuals explicit by giving them names and asserting their properties using concepts and roles.

Definition 2.4 (ABox)
Let $\mathsf{N_I}$ be a set of *individual names*. An *assertional axiom* is an expression of the form $C(x)$ (called *concept assertion*) or $r(x, y)$ (called *role assertion*), where $x, y \in \mathsf{N_I}$ are individual names, C a concept and r a role. A concept assertion is called *simple* if it is either of the form $A(x)$ or of the form $\neg A(x)$ (when negation is allowed in the language) where A is a concept name. A role assertion is called *simple* whenever it is of the form $r(x, y)$ with r a role name, i.e., not a complex role expressions. An *ABox* $\mathcal{A}$ is a finite set of assertional axioms. Here, $\mathcal{A}$ is called *simple* whenever all of its assertions are simple.

For the semantics, we require every interpretation additionally, to map each individual name $x \in \mathsf{N_I}$ to an element $x^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* a concept assertion $C(x)$ iff $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and it *satisfies* a role assertion $r(x, y)$ iff $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$. It *satisfies* an ABox $\mathcal{A}$ iff it satisfies every assertional axiom in $\mathcal{A}$. If such an interpretation $\mathcal{I}$ exists, then we say that $\mathcal{A}$ is *satisfiable* and we say that $\mathcal{I}$ is *model* of $\mathcal{A}$. ◇

In this work, we shall assume that all ABoxes are simple. The motivation is that ABoxes are viewed as data which is typically not expressed using arbitrary concept descriptions in many applications. This situation is similar in relational database theory where one never finds arbitrary (query) expressions as part of the data itself. Besides, with this assumption, terminological knowledge is strictly separated from assertional knowledge, and thus, the size of ABox is the measure of "raw" data involved in the applications [Hustadt *et al.*, 2005]. Finally, given a TBox and an ABox, one can combine them to obtain a description logic knowledge base.

Definition 2.5 (Knowledge Base)
A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{K}$ iff $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$. In this case, $\mathcal{K}$ is *satisfiable* and we say that $\mathcal{I}$ is a *model* of $\mathcal{K}$.

A concept C is *satisfiable w.r.t. a knowledge base* $\mathcal{K}$ iff there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$. A concept C is *subsumed* by a concept D *w.r.t. a knowledge base* $\mathcal{K}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. Two concepts C, D are *equivalent w.r.t. a knowledge base* $\mathcal{K}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. $\diamond$

From here on, given a knowledge base $\mathcal{K}$ (or a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$ depending on the context), $\mathcal{K} \models \varphi$ denotes the statement "every model of $\mathcal{K}$ satisfies φ" where φ can be a concept C, a statement of equivalence $C \equiv D$, a statement of subsumption $C \sqsubseteq D$ or an assertion ($C(x)$ or $r(x,y)$). When φ is the bottom-concept $\bot$, the statement is a shorthand for "$\mathcal{K}$ is unsatisfiable".

2.3 Reasoning Services and Complexity Measures

The purpose of a knowledge representation system based on description logics is not merely storing concept definitions and assertions [Baader & Nutt, 2003]. The semantics possessed by the knowledge base — comprising TBox and ABox — makes it equivalent to a set of axioms in first-order predicate logic. Hence, it contains implicit knowledge that can be made explicit through inferences. There are various inferences which form reasoning services of a DL-based knowledge representation system. Some of them which are considered standard in the literature [Baader & Nutt, 2003] are given below.

Definition 2.6 (Standard Reasoning Services)
Given a knowledge base $\mathcal{K}$, two concepts C and D, and an individual a, we call:

- *Concept satisfiability*, the problem of deciding whether C is satisfiable w.r.t. $\mathcal{K}$, i.e., whether $\mathcal{K} \not\models C \equiv \bot$.
- *(Concept) subsumption*, the problem of deciding whether C is subsumed by D w.r.t. $\mathcal{K}$, i.e., whether $\mathcal{K} \models C \sqsubseteq D$.
- *Knowledge base satisfiability (consistency)*, the problem of deciding whether $\mathcal{K}$ is satisfiable, i.e., whether $\mathcal{K} \not\models \bot$.

- *Instance checking*, the problem of deciding whether the assertion $C(a)$ is satisfied in every model of $\mathcal{K}$, i.e., whether $\mathcal{K} \models C(a)$. $\diamond$

In this thesis, we single out instance checking as the basic reasoning service. It is known that all other basic reasoning services can be reduced to instance checking. More precisely, for a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, two concepts C, D, and an individual a, we have

$$\begin{aligned}(\mathcal{T},\mathcal{A}) \not\models C \equiv \bot &\quad\Longleftrightarrow\quad (\mathcal{T},\mathcal{A} \cup \{C(a)\}) \not\models \bot(a)\\ (\mathcal{T},\mathcal{A}) \models C \sqsubseteq D &\quad\Longleftrightarrow\quad (\mathcal{T},\mathcal{A} \cup \{C(a)\}) \models D(a)\\ (\mathcal{T},\mathcal{A}) \not\models \bot &\quad\Longleftrightarrow\quad (\mathcal{T},\mathcal{A}) \not\models \bot(a)\end{aligned}$$

This means that concept satisfiability and knowledge base consistency can be reduced to the complement of instance checking, provided that the language can express the bottom concept. In addition, we also have that subsumption can be reduced to instance checking. It thus follows that instance checking is at least as hard as the other reasoning problems, even strictly harder in some DLs [Schaerf, 1993].

On the other hand, if the DL allows for negation, instance checking can be reduced to the complement of satisfiability, since $(\mathcal{T}, \mathcal{A}) \models C(a)$ iff $(\mathcal{T}, \mathcal{A} \cup \neg C(a))$ is unsatisfiable. One can also perform instance checking by combining a technique called abstraction together with subsumption. Abstraction consists of retrieving all concept and role assertions relevant to a and collecting them into a single concept. The instance checking $(\mathcal{T}, \mathcal{A}) \models C(a)$ is then solved by testing whether C subsumes the concept obtained from abstraction [Nebel, 1990a; Schaerf, 1993].

Another reasoning problem which is closely related to instance checking is the problem of *query answering* or *realization*. Given a knowledge base $\mathcal{K}$ and a concept C, query answering is the problem of finding all individual names x such that $x^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$. It thus is clear that query answering is essentially a finite set of instance checking tests. Moreover, instance checking can be seen as query answering with only one individual name. Hence, instance checking should be regarded as the central reasoning task for drawing conclusions upon individuals in knowledge bases.

For computational complexity, we use standard notions from complexity theory as presented in [Papadimitriou, 1994]. In particular, we will speak about the complexity classes P, NP, PSpace and ExpTime. The complexity class P (resp. PSpace, ExpTime) is the class of all decision problems that can be solved by a *deterministic* Turing Machine (TM) in polynomial time (resp. polynomial space, exponential time). NP is the class of all decision problems that can be solved by a *nondeterministic* TM in polynomial time. For a complexity class $\mathcal{C}$, the class co$\mathcal{C}$ is the set of problems whose complement is in $\mathcal{C}$.

Given a complexity class $\mathcal{C}$, a problem P_1 is said to be *$\mathcal{C}$-hard* if for every problem P_2 in $\mathcal{C}$, there is a reduction from P_2 to P_1 which is computable in polynomial time. A $\mathcal{C}$-hard problem is said to be *$\mathcal{C}$-complete* if it also belongs to $\mathcal{C}$.

The complexity of a problem is generally measured with respect to the size of its whole input. However, for instance checking, there is more than one piece of input that is given, namely the knowledge base which comprises the TBox and the ABox, the concept representing the query (*query concept*) and the individual. Here, the size of an individual is always constant, thus can be ignored. This leaves us with the size of TBox, the ABox and the query concept. Depending on which part of the input that dominates the overall size of the input, we may consider different kind of complexity measures, analogous to what has been suggested in [Vardi, 1986] for querying relational databases.

From here on, unless stated otherwise, we use $|\cdot|$ to denote the size function, i.e. the function which returns the size of its argument. In particular, the size of a concept C, a propositional formula φ, a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ are all defined as the number of symbols that are needed to write down C, φ, $\mathcal{T}$ and $\mathcal{A}$ respectively. For a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we define $|\mathcal{K}| = |\mathcal{T}| + |\mathcal{A}|$. For the instance checking problem, i.e., the problem of deciding whether $(\mathcal{T}, \mathcal{A}) \models C(a)$, we call:

- *data complexity*, the complexity of instance checking w.r.t. $|\mathcal{A}|$;
- *query complexity*, the complexity of instance checking w.r.t. $|C|$;
- *combined complexity*, the complexity of instance checking w.r.t. $|\mathcal{T}| + |\mathcal{A}| + |C| + |a| \approx |\mathcal{T}| + |\mathcal{A}| + |C|$.

There is a clear correspondence between querying DL knowledge base and querying relational database. The query concept corresponds to the query, the ABox corresponds to the data and the TBox corresponds to the database schema, i.e., the structure on which the data must obey [Hustadt *et al.*, 2005]. However in database terminology, complexity measures are only based on the size of the data and the size of the query, and no complexity measure is defined in terms of the schema. The reason is because the schema is not used when doing query answering in database systems. Therefore, we do not define a separate complexity measure of instance checking that is based on the size of the input TBox.

The majority of results on the complexity of reasoning in description logics deal with combined complexity. In particular, for the $\mathcal{EL}$-family, thorough studies can be found in [Baader, 2003b; Baader *et al.*, 2005a; Brandt, 2004b]. However, for applications in which the size of data (ABox) is much larger than both the size of the TBox and the size of the query concept, data complexity becomes a more realistic measure to the actual performance of a system. There are notably many applications where such assumption holds, for example, Enterprise Application Integration Systems, Data Integration Systems [Lenzerini, 2002], and the Semantic Web [Heflin & Hendler, 2001]. Motivated by this assumption, we thus consider data complexity for further investigations.

Chapter 3

Intractable Extensions of $\mathcal{EL}$ Regarding Data Complexity

In this chapter, we study the data complexity of instance checking for several extensions of $\mathcal{EL}$ with some common DL constructors, namely negation, disjunction, value restriction, unqualified number restrictions (only at-least restrictions, or only at-most restrictions or both) and some role constructors which include role negation, role union and transitive closures. For the extensions of $\mathcal{EL}$ with any of the aforementioned constructors, combined complexity of subsumption w.r.t. general TBoxes is ExpTime-complete [Baader *et al.*, 2005a]. In this chapter, we will show that for those extensions of $\mathcal{EL}$, the data complexity of instance checking is coNP-hard, and even coNP-complete in most cases. The aim here is to establish coNP-hardness results regarding data complexity of instance checking w.r.t. general TBoxes. However, except for one case of extension of $\mathcal{EL}$ with at-least restrictions, coNP-hardness for other cases can be established w.r.t. acyclic TBoxes, or even without TBoxes at all. Additionally, in order to establish coNP-completeness results for most cases, we refer to [Hustadt *et al.*, 2005] which established coNP upper bound (in fact, coNP-completeness) regarding data complexity of instance checking for DLs providing at least the constructors from the DL $\mathcal{ALC}$ and at most the constructors from the DL $\mathcal{SHIQ}$.

Note that Calvanese *et al.* [2006] already provided coNP-hardness results of data complexity of instance checking for some very simple DLs which are sublanguages of extensions of $\mathcal{EL}$ with negation, disjunction and value restriction. However, the coNP-hardness proofs in their paper require general TBoxes to be present. This is in contrast to our results where the coNP-hardness regarding data complexity for extensions of $\mathcal{EL}$ with negation, disjunction and value restriction require at most acyclic TBoxes.

We begin the chapter with a coNP-hardness (and thus coNP-completeness) result regarding data complexity for two extensions of $\mathcal{EL}$ with negation: atomic and full negation. As pointed out by Donini [2003], coNP-hardness may arise from a query language that can express both qualified existential restrictions and a pair of concepts such that their union is equivalent to the top-concept. In the extension of $\mathcal{EL}$ with atomic negation, such a pair of concepts are A and $\neg A$. Together with qualified existential restrictions, they can express a query concept that requires a sort of case analysis by the reasoner. The actual technique for showing this coNP-hardness result has been used in [Schaerf, 1993] to obtain coNP-hardness of instance checking regarding data complexity for $\mathcal{ALE}$ by a reduction from the complement of a variant of propositional satisfiability problem, named

2+2-SAT. In fact, we can directly use the same reduction to establish coNP-hardness of instance checking regarding data complexity for extensions of $\mathcal{EL}$ with negation because Schaerf's proof only uses qualified existential restriction, conjunction and negation which are all provided by $\mathcal{EL}$ plus negation.

The remaining part of the chapter builds upon Schaerf's idea to establish coNP-hardness (and coNP-completeness in most cases) of instance checking regarding data complexity for other extensions of $\mathcal{EL}$. The idea is to find a pair of concepts that behave like the pair A and $\neg A$ in the extension of $\mathcal{EL}$ with atomic negation, i.e., a pair of concepts whose union is equivalent to the top-concept. If such a pair of concepts can be found, then like in $\mathcal{EL}$ plus atomic negation, it suffices to use simple ABoxes and suitable query concepts (no TBoxes) to force a sort of case analysis by the reasoner. However, as we will see later, we may not always be able to find such a pair of concepts for every extension of $\mathcal{EL}$ considered in this chapter, and therefore, TBoxes may be needed in order to simulate the behavior of the pair A and $\neg A$ mentioned above.

3.1 Extensions of $\mathcal{EL}$ with Negation

Let $\mathcal{EL}^{\neg}$ be the extension of $\mathcal{EL}$ with negation and $\mathcal{EL}^{(\neg)}$ be obtained from $\mathcal{EL}^{\neg}$ by restricting the applicability of negation to concept names (*atomic* negation). We show that the data complexity of instance checking in $\mathcal{EL}^{(\neg)}$ is coNP-complete. The coNP upper bound is derived from the result of [Hustadt *et al.*, 2005] which showed that for knowledge bases containing simple ABoxes, data complexity of instance checking is coNP-complete for DLs which provide at least the constructors from $\mathcal{ALC}$ and at most the constructors from $\mathcal{SHIQ}$[1], regardless of the TBox formalisms used in the reasoning, i.e., the coNP lower bound (regarding data complexity) holds without TBoxes and the coNP upper bound (regarding data complexity) holds w.r.t. general TBoxes. Since $\mathcal{EL}^{(\neg)}$ is a sublanguage of $\mathcal{ALC}$ and $\mathcal{EL}^{\neg}$ is a notational variant of $\mathcal{ALC}$, the data complexity of instance checking for both extensions of $\mathcal{EL}$ is obviously also in coNP.

It thus remains to show the matching hardness result for $\mathcal{EL}^{(\neg)}$. Here, the coNP-hardness result will be shown to hold for knowledge bases without TBoxes, by a reduction from the complement of 2+2-SAT, a variant of the propositional satisfiability problem (SAT), to instance checking in $\mathcal{EL}^{(\neg)}$.

We first define the 2+2-SAT problem. Let $\mathcal{P}$ be a set of propositional variables and $\{true, false\}$ the set of propositional constants. A *literal* is either an element of $\mathcal{P} \cup \{true, false\}$ (*positive literal*) or negation of an element of $\mathcal{P} \cup \{true, false\}$ (*negative literal*). A *(2+2-literal) clause* is a disjunction of four literals: two positive and two negative ones. A *2+2-CNF formula* is a conjunction of 2+2-literal clauses.

A *truth assignment* is a function that assigns (*evaluates*) every element of the set $\mathcal{P} \cup \{true, false\}$ to an element of the set $\{0, 1\}$. Here, we use 0 and 1 to distinguish truth values from propositional constants. Let δ be a truth assignment. For propositional constants, we set $\delta(true) = 1$ and $\delta(false) = 0$. We also have δ evaluates a negative

[1]$\mathcal{SHIQ}$ is an extension of $\mathcal{ALC}$ with transitive roles, role hierarchy, inverse roles and qualified number restrictions, see [Baader *et al.*, 2003] for more detail of their syntax and semantics

literal $\neg p$ to 0 if $\delta(p) = 1$, and to 1 if $\delta(p) = 0$. For a 2+2-literal clause C, we define $\delta(C) = 1$ if at least one of the literals of C is evaluated to 1 by δ, otherwise, we define $\delta(C) = 0$. For a 2+2-CNF formula ψ, we define $\delta(\psi) = 1$ if every clause of ψ is evaluated to 1 by δ, otherwise, we define $\delta(\psi) = 0$. We also say that a clause (formula) is *satisfiable* if it is evaluated to 1 by some truth assignment, otherwise we say that the clause (formula) is *unsatisfiable*.

Definition 3.1 (2+2-SAT)
2+2-SAT is the problem of deciding whether a given 2+2-CNF formula is satisfiable. $\diamond$

The 2+2-SAT problem is a variant of the well-known 3-SAT problem which is NP-complete. In fact, a 3-literal clause mixing positive and negative literals can be transformed into a 2+2-literal clause by adding a fourth disjunct constantly false and an unmixed 3-literal clause can be replaced with two 2+2-literal clauses by adding a new variable such that satisfiability is preserved [Schaerf, 1993]. With this transformation, Schaerf has shown that 2+2-SAT is also NP-complete like 3-SAT.

Theorem 3.2 (Schaerf [1993])
2+2-SAT is NP*-complete.* $\diamond$

We now describe the reduction (also due to [Schaerf, 1993][2]) from the complement of 2+2-SAT to instance checking in $\mathcal{EL}^{(\neg)}$. In this reduction, it is shown that coNP-hardness holds even for an $\mathcal{EL}^{(\neg)}$-knowledge base without TBox, i.e., only ABox.

Let $\psi = C_1 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula with m propositional variables $p_1, \ldots, p_m$ and for every $i = 1, \ldots, n$, $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$ with $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, \ldots, p_m, true, false\}$. From ψ, we define an $\mathcal{EL}^{(\neg)}$ ABox $\mathcal{A}_\psi$ and a query concept Q. $\mathcal{A}_\psi$ has an individual p_j for each element of $\{p_1, \ldots, p_m, true, false\}$, an individual c_i for each clause C_i and an individual f for the whole formula ψ. $\mathcal{A}_\psi$ uses one primitive concept A and the following role names: Cl, P_1, $P2$, N_1, N_2.

$$
\begin{aligned}
\mathcal{A}_\psi \coloneqq \{&A(true), \neg A(false),\\
&Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n),\\
&P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}),\\
&\ldots\\
&P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-})\},\\
Q \coloneqq\ &\exists Cl.(\exists P_1.\neg A \sqcap \exists P_2.\neg A \sqcap \exists N_1.A \sqcap \exists N_2.A)
\end{aligned}
\tag{3.1}
$$

Intuitively, an individual q that is associated to an element of $\{p_1, \ldots, p_m, true, false\}$ belongs to A (resp. $\neg A$) iff its associated propositional variable or constant is evaluated to 1 (resp. 0). Note that $true$ only belongs to A and $false$ only belongs to $\neg A$. Now,

[2]Actually, the reduction in [Schaerf, 1993] was used to prove coNP-completeness of the DL $\mathcal{ALE}$ but uses only concept constructors that are actually available in $\mathcal{EL}^{(\neg)}$. Hence, it is perfectly suitable for our needs.

deciding whether $(\emptyset, \mathcal{A}_\psi) \models Q(f)$ corresponds to deciding whether the formula ψ is *unsatisfiable*, i.e., for every truth assignment of ψ, there exists a clause whose positive literals are evaluated to 0 and whose negative literals are evaluated to 1. The following claim is due to [Schaerf, 1993].

Claim: ψ is unsatisfiable if and only if $(\emptyset, \mathcal{A}_\psi) \models Q(f)$. $\diamond$

This claim and the fact that $|\mathcal{A}_\psi|$ is polynomial in $|\psi|$ yield the coNP-hardness of instance checking in $\mathcal{EL}^{(\neg)}$ regarding data complexity. This result also immediately applies to $\mathcal{EL}^{\neg}$ because $\mathcal{EL}^{\neg}$ is a superlanguage of $\mathcal{EL}^{(\neg)}$. Note that $\mathcal{EL}^{\neg}$ is a notational variant of the DL $\mathcal{ALC}$. As data complexity of instance checking for $\mathcal{ALC}$ is coNP-complete even w.r.t. general TBoxes [Hustadt *et al.*, 2005], it is clear that the coNP-completeness result in the following proposition which also applies for acyclic and general TBoxes.

Proposition 3.3
*Data complexity of instance checking for $\mathcal{EL}^{(\neg)}$ and $\mathcal{EL}^{\neg}$ with simple ABoxes and without TBoxes are co*NP*-complete.* $\diamond$

3.2 Extension of $\mathcal{EL}$ with Disjunction

Let $\mathcal{ELU}$ be the extension of $\mathcal{EL}$ with disjunction. We show that if we allow a TBox as a part of the input, data complexity of instance checking in $\mathcal{ELU}$ is coNP-hard. This result is, like the one for $\mathcal{EL}^{(\neg)}$, obtained by a reduction from the complement of 2+2-SAT.

In the $\mathcal{EL}^{(\neg)}$ case, negation is employed to obtain a pair of concepts A and $\neg A$ whose union is equivalent to the top-concept. Moreover, both concepts are disjoint. With this situation, deciding the instance checking problem requires a sort of case analysis, as one can associate those concepts precisely to truth assignments of propositional variables and constants in the 2+2-CNF formula ψ.

Now, in $\mathcal{ELU}$ case, we use two concept names A_T and A_F like the pair A and $\neg A$ in the $\mathcal{EL}^{(\neg)}$ case, i.e., an individual belongs to A_T (resp. A_F) if its associated propositional variable or constant is evaluated to 1 (resp. 0). Notice that, it suffices to consider only individuals that are associated to propositional variables and constants, since truth assignments of a 2+2-CNF formula depend only on the truth values of them. Thus, we need to ensure that $A_T \sqcup A_F$ holds on those individuals. If we allow a non simple ABox for the reduction, we can obviously modify the ABox in Equation (3.1) by replacing A with A_T and $\neg A$ with A_F, and using $A_T \sqcup A_F$ as concept assertions on every individual in it. But this cannot be done with a simple ABox and therefore, we need a TBox to associate $A_T \sqcup A_F$ either with the top-concept or with a new concept name.

For the TBox, if we use a general one, then we can just associate the disjunction of A_T and A_F with the top-concept using a single GCI $\top \sqsubseteq A_T \sqcup A_F$. Here though, the result would be stronger if we use an acyclic TBox. For the acyclic TBox, we introduce a new concept name, say A_{var}, and define it as $A_T \sqcup A_F$. For the ABox, we then use A_{var} in a concept assertion for every individual in the ABox that is associated with a propositional variable or constant. Notice that the concept definition $A_{var} \equiv A_T \sqcup A_F$

does not prevent the existence of a model of the input ABox and TBox such that an individual belongs to A_T and A_F simultaneously. To deal with this case, we modify the query concept by adding a disjunct representing this possibility. The reduction in a more formal form is in the following.

Let $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula where $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We assume that ψ has m propositional variables $p_1, \ldots, p_m$. We define an $\mathcal{ELU}$-ABox $\mathcal{A}_\psi$, an acyclic TBox $\mathcal{T}$ and a query concept Q as follows. $\mathcal{A}_\psi$ has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, and two individuals $true$ and $false$ for the corresponding propositional constants. In this reduction, the following concept names are used: A_{var}, A_T, A_F; and the following role names are used: Cl, P_1, $P2$, N_1, N_2, R. The ABox, TBox and query concept are described below.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{ & A_T(true), A_F(false),\\
& Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n),\\
& P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}),\\
& \ldots,\\
& P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-}),\\
& R(f, p_1), R(f, p_2), \ldots, R(f, p_m), R(f, true), R(f, false)\\
& A_{var}(p_1), A_{var}(p_2), \ldots, A_{var}(p_m), A_{var}(true), A_{var}(false)\},\\
\mathcal{T} \coloneqq \{ & A_{var} \equiv A_T \sqcup A_F\},\\
Q \coloneqq\ & \exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T) \sqcup \exists R.(A_T \sqcap A_F)
\end{aligned} \tag{3.2}$$

where $p_1, p_2, \ldots, p_m$ are propositional variables in ψ and for every $i = 1, \ldots, n$, it holds that $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \ldots, p_m, true, false\}$. Note that $A_T \sqcup A_F$ cannot be put into the ABox as the ABox is simple.

Claim: ψ is unsatisfiable if and only if $(\mathcal{T}, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: "$\Rightarrow$". Suppose ψ is unsatisfiable. Let $\mathcal{I}$ be a model of $(\mathcal{T}, \mathcal{A}_\psi)$. Then, $\{p_1^\mathcal{I}, \ldots, p_m^\mathcal{I}, true^\mathcal{I}, false^\mathcal{I}\} \subseteq (A_T \sqcup A_F)^\mathcal{I}$. If there is a $p \in \{p_1, \ldots, p_m, true, false\}$ such that $p^\mathcal{I} \in (A_T \sqcap A_F)^\mathcal{I}$, then since $\mathcal{I}$ is a model of $\mathcal{A}_\psi$, we have $(f^\mathcal{I}, p^\mathcal{I}) \in R^\mathcal{I}$ and thus obtain $f^\mathcal{I} \in (\exists R.(A_T \sqcap A_F))^\mathcal{I}$, i.e., $f^\mathcal{I} \in Q^\mathcal{I}$.

Otherwise, for every $q \in \{p_1, \ldots, p_m, true, false\}$, $q^\mathcal{I} \notin (A_T \sqcap A_F)^\mathcal{I}$. In this case, let $\delta_\mathcal{I}$ be the truth assignment such that $\delta_\mathcal{I}(q) = 1$ iff $q^\mathcal{I} \in A_T^\mathcal{I}$ and $\delta_\mathcal{I}(q) = 0$ iff $q^\mathcal{I} \in A_F^\mathcal{I}$. Since ψ is unsatisfiable, there is a clause $C_k = q_{k,1+} \vee q_{k,2+} \vee \neg q_{k,1-} \vee \neg q_{k,2-}$ that is not satisfied by $\delta_\mathcal{I}$. This implies $\delta_\mathcal{I}(q_{k,1+}) = \delta_\mathcal{I}(q_{k,2+}) = 0$ and $\delta_\mathcal{I}(q_{k,1-}) = \delta_\mathcal{I}(q_{k,2-}) = 1$. Thus, $q_{k,1+}^\mathcal{I}, q_{k,2+}^\mathcal{I} \in A_F^\mathcal{I}$ and $q_{k,1-}^\mathcal{I}, q_{k,2-}^\mathcal{I} \in A_T^\mathcal{I}$. Hence, we conclude $c_k^\mathcal{I} \in (\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)^\mathcal{I}$ from which it follows that $f^\mathcal{I} \in Q^\mathcal{I}$.

"$\Leftarrow$": Suppose ψ is satisfiable. We show that $(\mathcal{T}, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model of $\mathcal{T}$ and $\mathcal{A}_\psi$ that does not satisfy $Q(f)$. Let δ be a truth assignment satisfying ψ. We define an interpretation $\mathcal{I}_\delta$ as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_\delta} &= \{f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\} \\
x^{\mathcal{I}_\delta} &= x, \text{ for all } x \in \{f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\} \\
A_T^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 1, p \in \{true, p_1, \ldots, p_m\}\} \\
A_F^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 0, p \in \{false, p_1, \ldots, p_m\}\} \\
A_{var}^{\mathcal{I}_\delta} &= A_T^{\mathcal{I}_\delta} \cup A_F^{\mathcal{I}_\delta} \\
\rho^{\mathcal{I}_\delta} &= \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \in \Delta^{\mathcal{I}_\delta} \times \Delta^{\mathcal{I}_\delta} \mid \rho(x,y) \in \mathcal{A}_\psi\}, \text{ for } \rho \in \{Cl, P_1, P_2, N_1, N_2, R\}
\end{aligned}$$

It is easy to verify that $\mathcal{I}_\delta$ is a model of both $\mathcal{T}$ and $\mathcal{A}_\psi$. In addition, since ψ is satisfiable, for every clause C_i of ψ, at least one of its disjuncts is evaluated to 1, i.e., $\delta(q_{i,1+}) = 1$ or $\delta(q_{i,2+}) = 1$ or $\delta(q_{i,1-}) = 0$ or $\delta(q_{i,2-}) = 0$. Hence, for every individual $c_i^{\mathcal{I}_\delta}$, at least one of the following holds: $q_{i,1+}^{\mathcal{I}_\delta} \in A_T^{\mathcal{I}}$, $q_{i,2+}^{\mathcal{I}_\delta} \in A_T^{\mathcal{I}}$, $q_{i,1-}^{\mathcal{I}_\delta} \in A_F^{\mathcal{I}}$ or $q_{i,2-}^{\mathcal{I}_\delta} \in A_F^{\mathcal{I}}$. By definition of $\mathcal{I}_\delta$, we have $q_{i,1+}^{\mathcal{I}_\delta}$, $q_{i,2+}^{\mathcal{I}_\delta}$, $q_{i,1-}^{\mathcal{I}_\delta}$ and $q_{i,2-}^{\mathcal{I}_\delta}$ are respectively the only P_1-, P_2-, N_1-, and N_2-fillers of $c_i^{\mathcal{I}_\delta}$. We thus obtain that $c_i^{\mathcal{I}_\delta} \notin (\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)^{\mathcal{I}_\delta}$ for all $i = 1, \ldots, n$. Consequently, $f^{\mathcal{I}_\delta} \notin (\exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T))^{\mathcal{I}_\delta}$. Moreover, since every propositional variable and constant is never evaluated to both 1 and 0 simultaneously, we have $(A_T \sqcap A_F)^{\mathcal{I}_\delta} = \emptyset$. Thus, $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$, i.e., $\mathcal{I}_\delta$ does not satisfy $Q(f)$. $\diamond$

Observe that $|\mathcal{A}_\psi|$ is polynomial in $|\psi|$, and $|\mathcal{T}|$ and $|Q|$ do not depend on $|\psi|$. Hence, the claim implies that data complexity of instance checking in $\mathcal{ELU}$ w.r.t. acyclic TBoxes is coNP-hard. In addition, since data complexity of instance checking in $\mathcal{ALC}$ w.r.t. general TBoxes is in coNP [Hustadt *et al.*, 2005], and $\mathcal{ELU}$ is a sublanguage of $\mathcal{ALC}$, we obtain the following coNP-completeness result which also holds whenever the TBoxes are general.

Proposition 3.4
*Data complexity of instance checking in $\mathcal{ELU}$ w.r.t. simple ABoxes and acyclic TBoxes is co*NP*-complete.* $\diamond$

Notice that the proof of the proposition above requires a TBox as part of the input. What happens if the input of instance checking does not include a TBox? Let us now consider the problem of deciding $\mathcal{A} \models D(a)$ where $\mathcal{A}$ is a simple $\mathcal{ELU}$ ABox, D is an $\mathcal{ELU}$ concept and a is an individual. We show that this problem can decided in time polynomial in the size of $\mathcal{A}$.

First, note that if a does not appear in $\mathcal{A}$, the answer is trivially "no", i.e., $\mathcal{A} \not\models D(a)$, since for this case, one can easily construct a model $\mathcal{I}$ of $\mathcal{A}$ such that a belongs to no concept except the top-concept. Hence, we assume that a is one of the individuals in $\mathcal{A}$.

Let C be an $\mathcal{ELU}$ concept. We define $\mathsf{sub}(C)$, the set of subconcepts of an $\mathcal{ELU}$, as the smallest set S that contains C and has the following properties: if $C_1 \sqcup C_2 \in S$, then $\{C_1, C_2\} \in S$; if $C_1 \sqcap C_2 \in S$, then $\{C_1, C_2\} \in S$; and if $\exists r.C' \in S$, then $C' \in S$. The instance checking $\mathcal{A} \models D(a)$ can be decided in an algorithm given as follows. We use a (non-simple) ABox $\widehat{\mathcal{A}}$ as the data structure. We first compute the set $\mathsf{sub}(D)$ and initialize $\widehat{\mathcal{A}}$ to $\mathcal{A}$. We apply the following rules to $\widehat{\mathcal{A}}$ until no more rule is applicable (where d, d' are individuals occurring in $\widehat{\mathcal{A}}$):

(1) if $\top \in \mathsf{sub}(D)$ and $\top(d) \notin \widehat{\mathcal{A}}$, then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{\top(d)\}$;

(2) if $C_1 \sqcap C_2 \in \mathsf{sub}(D)$, $\{C_1(d), C_2(d)\} \subseteq \widehat{\mathcal{A}}$, and $(C_1 \sqcap C_2)(d) \notin \widehat{\mathcal{A}}$, then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{(C_1 \sqcap C_2)(d)\}$;

(3) if $C_1 \sqcup C_2 \in \mathsf{sub}(D)$, $\{C_1(d), C_2(d)\} \cap \widehat{\mathcal{A}} \neq \emptyset$, and $(C_1 \sqcup C_2)(d) \notin \widehat{\mathcal{A}}$, then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{(C_1 \sqcup C_2)(d)\}$;

(4) if $\exists r.C \in \mathsf{sub}(D)$, $\{r(d, d'), C(d')\} \subseteq \widehat{\mathcal{A}}$, and $(\exists r.C)(d) \notin \widehat{\mathcal{A}}$, then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{(\exists r.C)(d)\}$.

Let $\mathcal{A}_{out}$ be the ABox that is obtained by the application of the rules **(1)**–**(4)** to a simple $\mathcal{ELU}$ ABox $\mathcal{A}$ and an $\mathcal{ELU}$ query concept D. The algorithm answers "yes" if $a : D \in \mathcal{A}_{out}$ and "no" if $a : D \notin \mathcal{A}_{out}$. The fact that the algorithm terminates as well as the soundness and the completeness of the algorithm are shown by the following lemma.

Lemma 3.5
Let $\mathcal{A}$ be a simple $\mathcal{ELU}$ ABox, D an $\mathcal{ELU}$ concept. Assume that a is an individual name occurring in $\mathcal{A}$. Then both of the following hold.

1. *The rules **(1)**–**(4)** can only be applied a number of times polynomial in the size of $\mathcal{A}$, and each rule application can be done in time polynomial in the size of $\mathcal{A}$.*
2. *Let $\mathcal{A}_{out}$ be the ABox that is obtained by the application of the rules **(1)**–**(4)** to $\mathcal{A}$ and D. Then we have*
$$D(a) \in \mathcal{A}_{out} \text{ iff } \mathcal{A} \models D(a).$$

Proof.
1. First, the fact that each rule application can be done in time polynomial in the size of $\mathcal{A}$ is obvious. Next, note that the cardinality of $\mathsf{sub}(D)$ is linear in $|D|$. Each rule application adds a new concept assertion for an individual name whose concept is an element of $\mathsf{sub}(D)$. Since none of these rules removes assertions from $\mathcal{A}$, these rules can be performed on a particular individual names in a number of times linear in the size of $\mathsf{sub}(D)$. In addition, the number of individual names occurring in $\mathcal{A}$ is linear in the size of $\mathcal{A}$. Therefore, the total number of rule applications is linear in $|D||\mathcal{A}|$ which is linear in $|\mathcal{A}|$ because $|D|$ is considered constant.

2. We start with the "only if" direction, i.e., the soundness. Assume $D(a) \in \mathcal{A}_{out}$. We show that rule applications preserve modelship, i.e., for every rule, if $\mathcal{I}$ is a model of $\widehat{\mathcal{A}}$ before the rule is applied, then $\mathcal{I}$ is also a model of $\mathcal{A}$ after the rule is applied.

   (1): Trivial, since every individual belongs to the top-concept.

   (2): Let $\mathcal{I}$ be a model of $\widehat{\mathcal{A}}$ before the rule application. Thus, $d^{\mathcal{I}} \in C_1^{\mathcal{I}}$ and $d \in C_2^{\mathcal{I}}$, i.e., $d^{\mathcal{I}} \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} = (C_1 \sqcap C_2)^{\mathcal{I}}$. Since the rule adds the assertion $(C_1 \sqcap C_2)(d)$ to $\widehat{\mathcal{A}}$, we have that $\mathcal{I}$ is a model of $\widehat{\mathcal{A}}$ after the rule application.

(3): Let $\mathcal{I}$ be a model of $\widehat{\mathcal{A}}$ before the rule application. Thus, $d^{\mathcal{I}} \in C_1^{\mathcal{I}}$ or $d \in C_2^{\mathcal{I}}$, i.e., $d^{\mathcal{I}} \in C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} = (C_1 \sqcup C_2)^{\mathcal{I}}$. Since the rule adds the assertion $(C_1 \sqcup C_2)(d)$ to $\widehat{\mathcal{A}}$, we have that $\mathcal{I}$ is a model of $\widehat{\mathcal{A}}$ after the rule application.

(4): Let $\mathcal{I}$ be a model of $\widehat{\mathcal{A}}$ before the rule application. Thus, $(d^{\mathcal{I}}, (d')^{\mathcal{I}}) \in r^{\mathcal{I}}$ and $(d')^{\mathcal{I}} \in C^{\mathcal{I}}$. By the semantics, $d^{\mathcal{I}} \in (\exists r.C')^{\mathcal{I}}$. As this rule adds the assertion $(\exists r.C')(d)$ to $\widehat{\mathcal{A}}$, we have that $\mathcal{I}$ is a model of $\widehat{\mathcal{A}}$ after the rule application.

We now show that $\mathcal{A} \models D(a)$. Let $\mathcal{I}$ be a model of $\mathcal{A}$. Since rule applications preserve modelship, $\mathcal{I}$ is a model of $\mathcal{A}_{out}$. As $D(a) \in \mathcal{A}_{out}$, obviously $\mathcal{I}$ satisfies $D(a)$.

Next, we proceed with the "if" direction, i.e., the completeness. We will show that $D(a) \notin \mathcal{A}_{out}$ implies $\mathcal{A} \not\models D(a)$. Suppose $D(a) \notin \mathcal{A}_{out}$. Assume w.l.o.g. that $\{a_1, \ldots, a_n\}$ is the set of all individual names occurring in $\mathcal{A}$. We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ as follows:

- $\Delta^{\mathcal{J}} \coloneqq \{a_1, \ldots, a_n\}$, and $a_i^{\mathcal{J}} = a_i$ for $i = 1, \ldots, n$;
- $A^{\mathcal{J}} \coloneqq \{d \in \Delta^{\mathcal{J}} \mid A(d) \in \mathcal{A}_{out}\}$ for every concept name A;
- $r^{\mathcal{J}} \coloneqq \{(d, d') \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid r(d, d') \in \mathcal{A}_{out}\}$ for every role name r.

It is easy to see that $\mathcal{J}$ is a model of $\mathcal{A}_{out}$. Furthermore, it also a model of $\mathcal{A}$ because $\mathcal{A} \subseteq \mathcal{A}_{out}$. We now show that $a^{\mathcal{J}} \notin D^{\mathcal{J}}$ by induction on the structure of D. The induction base, i.e., for $D = A$, a concept name, is obvious by definition of $\mathcal{J}$. Assume for induction that the claim holds for some concepts C_1 and C_2. We distinguish cases based on the topmost constructor of D.

- $D = C_1 \sqcap C_2$. $(C_1 \sqcap C_2)(a) \notin \mathcal{A}_{out}$ implies $\{C_1(a), C_2(a)\} \nsubseteq \mathcal{A}_{out}$, because otherwise, the rule **(2)** would have been applicable to $\mathcal{A}_{out}$ which is not the case. Thus, $C_1(a) \notin \mathcal{A}_{out}$ or $C_2(a) \notin \mathcal{A}_{out}$. By induction hypothesis, we obtain that $a^{\mathcal{J}} \notin C_1^{\mathcal{J}}$ or $a^{\mathcal{J}} \notin C_2^{\mathcal{J}}$. Consequently, $a^{\mathcal{J}} \notin (C_1 \sqcap C_2)^{\mathcal{J}}$.
- $D = C_1 \sqcup C_2$. $(C_1 \sqcup C_2)(a) \notin \mathcal{A}_{out}$ implies $\{C_1(a), C_2(a)\} \cap \mathcal{A}_{out} = \emptyset$, because otherwise, the rule **(3)** would have been applicable to $\mathcal{A}_{out}$ which is not the case. Thus, $C_1(a) \notin \mathcal{A}_{out}$ and $C_2(a) \notin \mathcal{A}_{out}$. By induction hypothesis, we obtain that $a^{\mathcal{J}} \notin C_1^{\mathcal{J}}$ and $a^{\mathcal{J}} \notin C_2^{\mathcal{J}}$. Consequently, $a^{\mathcal{J}} \notin (C_1 \sqcup C_2)^{\mathcal{J}}$.
- $D = \exists r.C_1$. $(\exists r.C)(a) \notin \mathcal{A}_{out}$ implies $\{r(a, b), C_1(a)\} \nsubseteq \mathcal{A}_{out}$ for every individual name b occurring in $\mathcal{A}_{out}$, because otherwise, the rule **(4)** would have been applicable to $\mathcal{A}_{out}$ which is not the case. Thus, for every individual name b, $r(a, b) \notin \mathcal{A}_{out}$ or $C_1(a) \notin \mathcal{A}_{out}$. By induction hypothesis and definition of $\mathcal{J}$, we obtain that $(a^{\mathcal{J}}, b^{\mathcal{J}}) \notin r^{\mathcal{J}}$ or $a^{\mathcal{J}} \notin C_1^{\mathcal{J}}$. Consequently, $a^{\mathcal{J}} \notin (\exists r.C_1)^{\mathcal{J}}$.

This finishes the induction and we conclude that $\mathcal{J}$ is a model of $\mathcal{A}$ that does not satisfy $D(a)$, i.e., $\mathcal{A} \not\models D(a)$. ◇

The previous lemma provides with the termination, soundness and completeness of the algorithm for deciding the instance checking in $\mathcal{ELU}$ with simple ABoxes and without

TBoxes. In particular, the lemma shows that this decision problem can be solved in time polynomial in the size of the input ABox. Consequently, this yields the following proposition.

Proposition 3.6
Data complexity of instance checking for $\mathcal{ELU}$ w.r.t. simple ABoxes and without TBoxes is polynomial. ◇

3.3 Extensions of $\mathcal{EL}$ with Value Restriction

3.3.1 Adding Sink Restrictions: $\mathcal{EL}^{\forall r.\bot}$

We first look at the extension of $\mathcal{EL}$ named $\mathcal{EL}^{\forall r.\bot}$ which is $\mathcal{EL}$ extended with sink restrictions $\forall r.\bot$. For $\mathcal{EL}^{\forall r.\bot}$, data complexity of instance checking is coNP-hard even without TBoxes. The coNP-hardness is obtained by a reduction from the complement of 2+2-SAT. The reduction is similar to the one for $\mathcal{EL}^{(\neg)}$ given in (3.1) where instead of using the pair A and $\neg A$, we use the pair of concepts $\exists s.\top$ and $\forall s.\bot$. The whole reduction is given below.

Let $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula where $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We assume that ψ has m propositional variables $p_1, \ldots, p_m$. We define a simple ABox $\mathcal{A}_\psi$, and an $\mathcal{EL}^{\forall r.\bot}$ query concept Q as follows. $\mathcal{A}_\psi$ has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, and two individuals $true$ and $false$ for the corresponding propositional constants. In this reduction, the following role names are used: Cl, P_1, $P2$, N_1, N_2, and S. It also holds that for every $1 \leq i \leq n$, $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, \ldots, p_m, true, false\}$. The ABox, and query concept are described below.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{ & Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n), \\
& P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}), \\
& \ldots \\
& P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-})\}, \\
Q \coloneqq \; & \exists Cl.(\exists P_1 \forall S.\bot \sqcap \exists P_2 \forall S.\bot \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)
\end{aligned} \tag{3.3}$$

It can easily be verified that $(\exists S.\top)^{\mathcal{I}} \cup (\forall S.\bot)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $(\exists S.\top)^{\mathcal{I}} \cap (\forall S.\bot)^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{A}_\psi$. This clearly simulates the behavior of the pair A and $\neg A$ in $\mathcal{EL}^{(\neg)}$. Thus, analogous with the reduction from the complement of 2+2-SAT to instance checking in $\mathcal{EL}^{(\neg)}$, we have that ψ is unsatisfiable iff $(\emptyset, \mathcal{A}_\psi) \models Q(f)$. This establish the coNP lower bound of data complexity of instance checking in $\mathcal{EL}^{\forall r.\bot}$ with simple ABoxes and without TBoxes. The upper bound is obtained from the fact that in $\mathcal{ALC}$, the superlanguage of $\mathcal{EL}^{\forall r.\bot}$, data complexity of instance checking w.r.t. general TBoxes is in coNP [Hustadt *et al.*, 2005]. Hence, we establish the following coNP-completeness result which also holds whenever the TBoxes are acyclic or general.

Proposition 3.7
*Data complexity of instance checking in $\mathcal{EL}^{\forall r.\bot}$ w.r.t. simple ABoxes and without TBoxes is co*NP*-complete.* ◇

3.3.2 Adding Value Restriction without The Bottom-Concept: $\mathcal{EL}^{\forall}$

Next, we consider $\mathcal{EL}^{\forall}$, the extension of $\mathcal{EL}$ with value restriction. Note that $\mathcal{EL}^{\forall}$ does not provide the bottom-concept. But even without using bottom, adding value restriction to $\mathcal{EL}$ makes data complexity of instance checking w.r.t. acyclic TBoxes intractable. This is proved by a reduction from the complement of 2+2-SAT that is similar to the one for $\mathcal{EL}^{\forall r.\bot}$. Here, we employ a pair of concepts $\exists s.\top$ and $\forall s.A$ in a way similar to the way we used the pair $\exists s.\top$ and $\forall s.\bot$ in the $\mathcal{EL}^{\forall r.\bot}$ case. It is easy to see that the union of $\exists s.\top$ and $\forall s.A$ is equivalent to the top-concept, although unlike the pair A and $\neg A$ in $\mathcal{EL}^{(\neg)}$, $\exists s.\top$ and $\forall s.A$ are not necessarily disjoint. In $\mathcal{ELU}$, this is solved by using disjunction in the query concept. But this cannot be done here, since $\mathcal{EL}^{\forall}$ can express neither negation nor disjunction.

To solve the above problem, we add an additional individual d to the ABox. It performs as a marker that "recognizes" any individuals belonging to both $\exists S.\top$ and $\forall S.A$ simultaneously. The aim is that whenever in a model of $\mathcal{A}_\psi$, the concept $\exists S.\top \sqcap \forall S.A$ is true at an individual that is associated to a propositional variable or constant occurring in the 2+2-CNF formula ψ, the query assertion $Q(f)$ where $Q \coloneqq \exists Cl.(\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S\top \sqcap \exists N_2 \exists S.\top)$ can still be satisfied. This marking mechanism is accomplished by first connecting d through all roles P_1, P_2, N_1 and N_2 to all individuals that represents propositional variables and constants and then connecting the individual representing the whole formula (on which the query concept is considered) to d via the role Cl. The reduction is presented in a more formal form in the following.

Let $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula where $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We assume that ψ has m propositional variables $p_1, \ldots, p_m$. We define a simple ABox $\mathcal{A}_\psi$, and an $\mathcal{EL}^{\forall}$ query concept Q as follows. $\mathcal{A}_\psi$ has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, two individuals $true$ and $false$ for the corresponding propositional constants and one additional individual d. In this reduction, we use a concept name A; and role names: Cl, P_1, $P2$, N_1, N_2, and S.

$$
\begin{aligned}
\mathcal{A}_\psi \coloneqq \{ & Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n), Cl(f, d), \\
& P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}), \\
& \ldots, \\
& P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-}), \\
& P_1(d, p_1), P_2(d, p_1), N_1(d, p_1), N_2(d, p_1), \\
& \ldots, \\
& P_1(d, p_m), P_2(d, p_m), N_1(d, p_m), N_2(d, p_m), \\
& P_1(d, true), P_2(d, true), N_1(d, true), N_2(d, true), \\
& P_1(d, false), P_2(d, false), N_1(d, false), N_2(d, false)\}, \\
Q \coloneqq \ & \exists Cl.(\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)
\end{aligned}
\tag{3.4}
$$

where $\{p_1, p_2, \ldots, p_m\}$ is the set of all propositional variables in ψ and for all $i = 1, \ldots, n$, it holds that $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \ldots, p_m, true, false\}$.

Claim: ψ is unsatisfiable if and only if $(\emptyset, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: For the "only if" part, assume that ψ is unsatisfiable. Note that for every interpretation $\mathcal{J}$, $\Delta^{\mathcal{J}} = (\exists S.\top)^{\mathcal{J}} \cup (\forall S.A)^{\mathcal{J}}$. Let $\mathcal{I}$ be a model of $\mathcal{A}_\psi$. Hence, in particular, for every $q \in \{p_1, \dots, p_m, true, false\}$, $q^{\mathcal{I}} \in (\exists S.\top)^{\mathcal{I}}$ or $q^{\mathcal{I}} \in (\forall S.A)^{\mathcal{I}}$.

If there is a $q \in \{p_1, \dots, p_m, true, false\}$ such that $q^{\mathcal{I}} \in (\exists S.\top)^{\mathcal{I}} \cap (\forall S.A)^{\mathcal{I}}$, then since $q^{\mathcal{I}}$ is a P_1-, P_2-, N_1- and N_2-filler of $d^{\mathcal{I}}$, it holds that $d^{\mathcal{I}} \in (\exists P_1 \forall S.A)^{\mathcal{I}} \cap (\exists P_2 \forall S.A)^{\mathcal{I}} \cap (\exists N_1 \exists S.\top)^{\mathcal{I}} \cap (\exists N_2 \exists S.\top)^{\mathcal{I}} = (\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)^{\mathcal{I}}$. Hence, since $d^{\mathcal{I}}$ is a Cl-filler of $f^{\mathcal{I}}$, we obtain $f^{\mathcal{I}} \in (\exists Cl.(\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top))^{\mathcal{I}}$, i.e., $f^{\mathcal{I}} \in Q^{\mathcal{I}}$.

Otherwise, if for every $q \in \{p_1, \dots, p_m, true, false\}$, it holds that $q^{\mathcal{I}} \notin (\exists S.\top)^{\mathcal{I}} \cap (\forall S.A)^{\mathcal{I}}$, then either $q^{\mathcal{I}} \in (\exists S.\top)^{\mathcal{I}}$ or $q^{\mathcal{I}} \in (\forall S.A)^{\mathcal{I}}$ but not both. Let $\delta_{\mathcal{I}}$ be a truth assignment to all propositional variable q of ψ such that $\delta_{\mathcal{I}}(q) = 1$ iff $q^{\mathcal{I}} \in (\exists S.\top)^{\mathcal{I}}$ and $\delta_{\mathcal{I}}(q) = 0$ iff $q^{\mathcal{I}} \in (\forall S.A)^{\mathcal{I}}$. Since ψ is unsatisfiable, there is a clause C_i which is not satisfied by $\delta_{\mathcal{I}}$. Thus, $\delta_{\mathcal{I}}(q_{1+,i}) = \delta_{\mathcal{I}}(q_{2+,i}) = 0$ and $\delta_{\mathcal{I}}(q_{1-,i}) = \delta_{\mathcal{I}}(q_{2-,i}) = 1$. This implies that P_1- and P_2-fillers of $c_i^{\mathcal{I}}$ belong to $(\forall S.A)^{\mathcal{I}}$ and N_1- and N_2-fillers of $c_i^{\mathcal{I}}$ belong to $(\exists S.\top)^{\mathcal{I}}$. Hence, we conclude that $c_i^{\mathcal{I}} \in (\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)^{\mathcal{I}}$. Since $(f^{\mathcal{I}}, c_i^{\mathcal{I}}) \in Cl^{\mathcal{I}}$, consequently $f^{\mathcal{I}} \in (\exists Cl.(\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top))^{\mathcal{I}}$, i.e., $f^{\mathcal{I}} \in Q^{\mathcal{I}}$.

For the "if" part, suppose ψ is satisfiable. We show that $(\emptyset, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model of $\mathcal{A}_\psi$ that does not satisfy $Q(f)$. Let δ be a truth assignment that satisfies ψ. We define an interpretation $\mathcal{I}_\delta$ as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_\delta} &= \{d, f, c_1, \dots, c_n, p_1, \dots, p_m, true, false\} \\
x^{\mathcal{I}_\delta} &= x, \text{ for all } x \in \{d, f, c_1, \dots, c_n, p_1, \dots, p_m, true, false\} \\
A^{\mathcal{I}_\delta} &= \emptyset \\
S^{\mathcal{I}_\delta} &= \{(p^{\mathcal{I}_\delta}, true) \mid \delta(p) = 1, p \in \{true, p_1, \dots, p_m\}\} \\
\rho^{\mathcal{I}_\delta} &= \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \mid \rho(x, y) \in \mathcal{A}_\psi\} \quad \text{for } \rho \in \{Cl, P_1, P_2, N_1, N_2\}
\end{aligned}$$

It is straightforward to verify that $\mathcal{I}_\delta$ is a model of $\mathcal{A}_\psi$. We now show $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$. Note that since ψ is satisfiable, for every clause C_i of ψ, at least one of its disjuncts is evaluated to 1, i.e., $\delta(q_{i,1+}) = 1$ or $\delta(q_{i,2+}) = 1$ or $\delta(q_{i,1-}) = 0$ or $\delta(q_{i,2-}) = 0$. Because $\Delta^{\mathcal{I}} = (\exists S.\top)^{\mathcal{I}} \cup (\forall S.A)^{\mathcal{I}}$ also holds, we have that for every individual $c_i^{\mathcal{I}_\delta}$, at least one of the following holds: $q_{i,1+}^{\mathcal{I}_\delta} \in (\exists S.\top)^{\mathcal{I}}$, $q_{i,2+}^{\mathcal{I}_\delta} \in (\exists S.\top)^{\mathcal{I}}$, $q_{i,1-}^{\mathcal{I}_\delta} \in (\forall S.A)^{\mathcal{I}}$ or $q_{i,2-}^{\mathcal{I}_\delta} \in (\forall S.A)^{\mathcal{I}}$. By definition of $\mathcal{I}_\delta$, $q_{i,1+}^{\mathcal{I}_\delta}$, $q_{i,2+}^{\mathcal{I}_\delta}$, $q_{i,1-}^{\mathcal{I}_\delta}$ and $q_{i,2-}^{\mathcal{I}_\delta}$ are respectively the only P_1-, P_2-, N_1-, and N_2-filler of $c_i^{\mathcal{I}_\delta}$. Thus, we obtain $c_i^{\mathcal{I}_\delta} \notin (\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)^{\mathcal{I}_\delta}$ for all $i = 1, \dots, n$. Finally, we conclude $f^{\mathcal{I}_\delta} \notin (\exists Cl.(\exists P_1 \forall S.A \sqcap \exists P_2 \forall S.A \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top))^{\mathcal{I}_\delta}$, i.e., $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$. $\diamond$

Note that in the reduction above, $|\mathcal{A}_\psi|$ is polynomial in $|\psi|$. Moreover, $|Q|$ does not depend on $|\psi|$. Consequently, by the previous claim, we obtain the following the coNP lower bound of data complexity of instance checking in $\mathcal{EL}^{\forall}$ with simple ABoxes and without TBoxes. The upper bound is obtained from the fact that in $\mathcal{ALC}$, the superlanguage of $\mathcal{EL}^{\forall}$, data complexity of instance checking w.r.t. general TBoxes is in coNP [Hustadt *et al.*, 2005]. Hence, we establish the following proposition. Note that this proposition also applies whenever the TBoxes are acyclic or general.

Proposition 3.8
*Data complexity of instance checking in $\mathcal{EL}^{\forall}$ w.r.t. simple ABoxes and without TBoxes is co*NP*-complete.* $\diamond$

3.4 Extensions of $\mathcal{EL}$ with Unqualified Number Restrictions

3.4.1 Adding Both At-Most and At-Least Restrictions, and Adding Only At-Most Restrictions: $\mathcal{EL}^{\leq k_1, \geq k_2}$, $\mathcal{EL}^{\leq k}$ and $\mathcal{EL}^{kf}$

We start with the DLs $\mathcal{EL}^{\leq k_1, \geq k_2}$. These are DLs which are obtained by extending $\mathcal{EL}$ with both at-most restrictions $(\leq k_1\ r)$ and at-least restrictions $(\geq k_2\ r)$ where k_1 and k_2 are fixed nonnegative integers.

For these DLs, the constructor $(\geq k_2\ r)$ is apparently not needed in establishing coNP-hardness regarding data complexity of instance checking if the constructor $(\leq k_1\ r)$ is used. In other words, for fixed nonnegative integers k_1 and k_2, the reduction from the complement of 2+2-SAT to instance checking in $\mathcal{EL}^{\leq k_1, \geq k_2}$ is *precisely* the same as the reduction from the complement of 2+2-SAT to instance checking in $\mathcal{EL}^{\leq k_1}$, DLs which are obtained by extending $\mathcal{EL}$ with at-most restrictions $(\leq k_1\ r)$ only (see the reduction on page 27). The reason is because $(\geq 0\ r) \equiv \top$ and for every integers $k_2 \geq 1$, $(\geq k_2\ r)$ is subsumed by $\exists r.\top$, a constructor that is already provided by $\mathcal{EL}$. Of course, besides using the reduction mentioned previously, one could also obtained the coNP-hardness result of data complexity of instance checking in $\mathcal{EL}^{\leq k_1, \geq k_2}$ as a direct consequence from the coNP-hardness result of data complexity in $\mathcal{EL}^{\leq k_1}$, since for fixed nonnegative integers k_1 and k_2, $\mathcal{EL}^{\leq k_1, \geq k_2}$ is a superlanguage of $\mathcal{EL}^{\leq k_1}$. Because of these reasons, we will now just proceed to the extensions of $\mathcal{EL}$ with at-most restrictions only.

For DLs $\mathcal{EL}^{\leq k}$, extensions of $\mathcal{EL}$ with at-most restriction $(\leq k\ r)$ with k a fixed nonnegative integer, we first look at the case where $k = 0$. Obviously, $(\leq 0\ r) \equiv \forall r.\bot$. This means that the DL $\mathcal{EL}^{\leq 0}$ is actually a notational variant of $\mathcal{EL}^{\forall r.\bot}$ for which Proposition 3.7 holds.

For $k > 0$, the constructor $(\leq k\ r)$ introduces a form of functionality to the language. Functionality originally refers to the case in which a particular role is required to have at most one filler. In this subsection, we introduce k-functionality as a generalization of the notion of functionality in the sense that the number of fillers of a role is limited to k, instead of one. In addition, depending on whether k-functionality is required everywhere or not, we distinguish two kinds of k-functionality: global and local.

Definition 3.9
Let r be a role and a an individual. We say that r is *k-functional on a* if a has at most k r-fillers. For the special case where $k = 1$, we simply say that r is *functional on a*.

Let $\mathcal{I}$ be an interpretation with $\Delta^{\mathcal{I}}$ as its domain. Then we say that r is *locally k-functional* if r is k-functional on some individuals in $\Delta^{\mathcal{I}}$, and we say that r is *globally k-functional* if r is k-functional on every individual in $\Delta^{\mathcal{I}}$. $\diamond$

It is clear from the definition that local k-functionality implies global k-functionality, but not conversely. Second, observe that if a role r is locally k-functional, then it may behave as an ordinary binary relation on some individuals in the domain of interpretation.

On syntactic level, $\mathcal{EL}$ with global k-functionality restricts the use of at-most constructors $(\leq k\ r)$ to GCIs of the form $\top \sqsubseteq (\leq k\ r)$. In case GCIs are not allowed in the knowledge base (and thus the constructor $(\leq k\ r)$ cannot be explicitly expressed anywhere in the knowledge base and the query concept), we can just divide the set of all roles into two disjoint sets where the first set contains roles interpreted as ordinary binary relation and the second set contains roles interpreted as k-functions, i.e., binary relations with at most k fillers. On the other hand, $\mathcal{EL}$ with local k-functionality imposes no restriction on how or where the constructor $(\leq k\ r)$ is used, i.e., this constructor is used like other DL constructors would be used.

In the current section, for some fixed nonnegative integer k, the DLs $\mathcal{EL}^{\leq k}$ are extensions of $\mathcal{EL}$ with local k-functionality, whereas the DLs $\mathcal{EL}^{kf}$ are extensions of $\mathcal{EL}$ with global k-functionality. For example, $\mathcal{EL}^{\leq 1}$ and $\mathcal{EL}^{\leq 2}$ are extensions of $\mathcal{EL}$ with local functionality and local 2-functionality, respectively. Meanwhile, $\mathcal{EL}^{1f}$ is the extension of $\mathcal{EL}$ with global 1-functionality and also known simply as $\mathcal{EL}^{f}$, the extension of $\mathcal{EL}$ with global functionality; and $\mathcal{EL}^{2f}$ is the extension of $\mathcal{EL}$ with global 2-functionality.

Instance Checking for $\mathcal{EL}^{\leq k}$

We now show that data complexity of instance checking is coNP-hard for $\mathcal{EL}^{\leq k}$ where $k > 0$. We use the fact that the union of $\exists r.\top$ and $(\leq k\ r)$ is equivalent to the top-concept. Note that they are not necessarily disjoint, i.e., $\exists r.\top \sqcap (\leq k\ r)$ is satisfiable. The reduction is very similar to the $\mathcal{EL}^{\forall}$ case and presented in the following.

Let $\psi = C_1 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula where $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We assume that ψ has m propositional variables $p_1, \ldots, p_m$. We define a simple ABox $\mathcal{A}_\psi$, and an $\mathcal{EL}^{\leq k}$ query concept Q where $k > 0$ as follows. $\mathcal{A}_\psi$ has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, two individuals $true$ and $false$ for the corresponding propositional constants and one additional individual d which is used for marking mechanism like the one that was described in the $\mathcal{EL}^{\forall}$ case (Subsection 3.3.2). In this reduction, we use role names: Cl, P_1, $P2$, N_1, N_2, and S.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{ & Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n), Cl(f, d), \\
& P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}), \\
& \ldots, \\
& P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-}), \\
& P_1(d, p_1), P_2(d, p_1), N_1(d, p_1), N_2(d, p_1), \qquad (3.5) \\
& \ldots,
\end{aligned}$$

$$\begin{aligned}
&P_1(d,p_m), P_2(d,p_m), N_1(d,p_m), N_2(d,p_m),\\
&P_1(d,true), P_2(d,true), N_1(d,true), N_2(d,true),\\
&P_1(d,false), P_2(d,false), N_1(d,false), N_2(d,false)\},\\
Q \coloneqq\ &\exists Cl.(\exists P_1.(\leq k\ S) \sqcap \exists P_2.(\leq k\ S) \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)
\end{aligned}$$

where for all $i = 1,\dots,n$, $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \dots, p_m, true, false\}$.

Claim: ψ is unsatisfiable if and only if $(\emptyset, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: The "only if" part is similar to the one for $\mathcal{EL}^\forall$ on page 25. Assume that ψ is unsatisfiable. Note that for every model $\mathcal{J}$ of $\mathcal{A}_\psi$, $(\exists S.\top)^\mathcal{J} \cup (\leq k\ S)^\mathcal{I} = \Delta^\mathcal{J}$. Let $\mathcal{I}$ be a model of $\mathcal{A}_\psi$. If there is an individual name $q \in \{p_1,\dots,p_m, true, false\}$ which satisfies $q^\mathcal{I} \in ((\exists S.\top) \sqcap (\leq k\ S))^\mathcal{I}$, i.e., $q^\mathcal{I}$ has k' S-fillers, $1 \leq k' \leq k$, then we obtain that $d^\mathcal{I} \in (\exists P_1.(\leq k\ S) \sqcap \exists P_2.(\leq k\ S) \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)^\mathcal{I}$. This yields $f^\mathcal{I} \in (\exists Cl.(\exists P_1.(\leq k\ S) \sqcap \exists P_2.(\leq k\ S) \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top))^\mathcal{I} = Q^\mathcal{I}$. Otherwise, for every $q \in \{p_1,\dots,p_m, true, false\}$, either $q^\mathcal{I} \in (\exists S.\top)^\mathcal{I}$ or $p^\mathcal{I} \in (\leq k\ S)^\mathcal{I}$ but not both. In this case, unsatisfiability of ψ implies $f^\mathcal{I} \in (\exists Cl.(\exists P_1.(\leq k\ S) \sqcap \exists P_2.(\leq k\ S) \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top))^\mathcal{I} = Q^\mathcal{I}$ via one of $c_i^\mathcal{I}$, $i \in \{1,\dots,n\}$.

For the "if" part, suppose ψ is satisfiable and δ is a truth assignment that satisfies ψ. We show that $(\emptyset, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model of $\mathcal{A}_\psi$ that does not satisfy $Q(f)$. We define an interpretation $\mathcal{I}_\delta$ as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_\delta} &= \{d, f, c_1,\dots,c_n, p_1,\dots,p_m, true, false\}\\
x^{\mathcal{I}_\delta} &= x,\ \text{for all } x \in \{d, f, c_1,\dots,c_n, p_1,\dots,p_m, true, false\}\\
S^{\mathcal{I}_\delta} &= \{(p^{\mathcal{I}_\delta}, true) \mid \delta(p) = 1, p \in \{true, p_1,\dots,p_m\}\}\\
\rho^{\mathcal{I}_\delta} &= \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \mid \rho(x,y) \in \mathcal{A}_\psi\} \quad \text{for } \rho \in \{Cl, P_1, P_2, N_1, N_2\}
\end{aligned}$$

It easy to verify that $\mathcal{I}_\delta$ is indeed a model of $\mathcal{A}_\psi$. Moreover, it is straightforward to show that satisfiability of ψ implies $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$ (see the proof of the claim that leads to Proposition 3.8). $\diamond$

It is obvious that $|\mathcal{A}_\psi|$ is polynomial in $|\psi|$, and $|Q|$ do not depend on $|\psi|$. Hence, the previous claim yields the coNP-hardness regarding data complexity of instance checking with simple ABoxes and without TBoxes in $\mathcal{EL}^{\leq k_1, \geq k_2}$ and $\mathcal{EL}^{\leq k}$. In addition, since $\mathcal{EL}^{\leq k_1, \geq k_2}$ and $\mathcal{EL}^{\leq k}$ are sublanguages of the DL $\mathcal{SHIQ}$, and data complexity of instance checking in $\mathcal{SHIQ}$ w.r.t. simple ABoxes is in coNP [Hustadt *et al.*, 2005], we conclude the following proposition. Note that the following proposition also holds whenever the TBoxes are acyclic or even general.

Proposition 3.10
*For all fixed nonnegative integers k, k_1 and k_2, data complexity of instance checking in $\mathcal{EL}^{\leq k_1, \geq k_2}$ and $\mathcal{EL}^{\leq k}$ w.r.t. simple ABoxes and without TBoxes are co*NP*-complete.* $\diamond$

Note that for the cases where $k_2 = k_1 + 1$, coNP-hardness result of data complexity in $\mathcal{EL}^{\leq k_1, \geq k_2}$ can also be obtained by observing the fact that the pair $(\leq k_1\ r)$ and $(\geq (k_1+1)\ r)$ behaves precisely like the pair A and $\neg A$ in $\mathcal{EL}^{(\neg)}$.

Instance Checking for $\mathcal{EL}^{kf}$

The previous result provided us with coNP-completeness of instance checking with simple ABoxes for $\mathcal{EL}$ extended with local k-functionality (regarding data complexity). Now one could also ask whether an analogous coNP-completeness result can also be obtained if $\mathcal{EL}$ is extended with global k-functionality. Note that in this case, it suffices to establish a matching coNP-hardness result.

As we will see in the following, we can obtain coNP-hardness regarding data complexity of instance checking in $\mathcal{EL}^{kf}$ when $k > 1$ by employing a reduction from the complement of 2+2-SAT. For $\mathcal{EL}^{1f}$, data complexity of instance checking w.r.t. general TBoxes is polynomial and will be discussed in more detail in Chapter 4.

Here, to express global k-functionality, instead of using GCIs, we assume that there are two sets of role names. One of them consists of role names which are globally k-functional and the other consists of role names interpreted in the usual way. We also use $\exists F^k.C$ as an abbreviation for

$$\underbrace{\exists F \ldots \exists F}_{k}.C$$

For the reduction from the complement of 2+2-SAT, we want to find a pair of concepts that behave like the pair $\exists r.\top$ and $\forall r.A$ in the $\mathcal{EL}^{\forall}$ case. The idea stems from the following observation. Whenever an individual name p has $k+1$ F-fillers and the role name F is globally k-functional, then two of these $k+1$ fillers must be identified. Such an identification may force a sort of case analysis because there are $\frac{k(k+1)}{2}$ possible pairs of individual names for identification. For the reduction from the complement of 2+2-SAT, we only need two cases (corresponding to two truth values). We thus distinguish one individual name b from the other individual names, and derive the following two cases. First case is where b is identified with one of the other individual names, whereas the second case is whenever the identification does not involve b, i.e., occurs between two individual names different from b. This idea can be realized using ABox assertions described below where we also ensure that there are initially $k+1$ F-fillers of p.

$$\begin{aligned}
&F(p,a_1),\ldots,F(p,a_k),F(p,b),\\
&A(a_1),\ldots,A(a_k),B(b),\\
&F(a_1,a_2),F(a_1,a_3),\ldots,F(a_1,a_k),\\
&F(a_2,a_3),\ldots,F(a_2,a_k),\\
&\ldots,\\
&F(a_{k-1},a_k)
\end{aligned}$$

Note that every pair of individual names from $a_1,\ldots,a_k$ are also connected through F, and each of these individual names has at most $k-1$ F-fillers. Now, since F is globally k-functional, two individual names from $a_1,\ldots,a_k,b$ must be identified. If b is identified with one of a_i then the concept $\exists F.(A \sqcap B)$ is true at p. Otherwise, two individual names a_i and a_j from $a_1,\ldots,a_k$ are identified which implies F connects a_i to itself (a "loop" occurs). In this case, the concepts $\exists F^n.\top$ for $n \geq 1$ are true at p. Note that for $1 \leq n \leq k$, $\exists F^n.\top$ is always true at p. Hence, the pair of concepts $\exists F^{k+1}.\top$ and $\exists F.(A \sqcap B)$ are the pair that we were looking for.

Observe that it suffices to apply the trick described above to individual names representing propositional variables in a 2+2-CNF formula ψ. More precisely, the individual name p used in the trick is intended to be those individual names representing propositional variables. Moreover, the trick does not exclude the case that occurs whenever an individual (which corresponds to a propositional variable or constant) belongs to both $\exists F^{k+1}.\top$ and $\exists F.(A \sqcap B)$. To deal with such a case, we employ a marking mechanism using a new individual name d like the one for $\mathcal{EL}^{\forall}$.

We now give a formal reduction from the complement of 2+2-SAT to instance checking for $\mathcal{EL}^{kf}$, $k > 1$. In this reduction, the following concept names are used: A and B; and the following role names are used: Cl, P_1, $P2$, N_1, N_2, F. We assume that F is a globally k-functional role. We start with a 2+2-CNF formula $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ with m propositional variables and n clauses where each clause is of the form $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We translate ψ to a simple ABox $\mathcal{A}_\psi$, and a query concept Q which are described below. The simple ABox $\mathcal{A}_\psi$ contains $k+2$ individual names $p_j, a_{j,1}, \ldots, a_{j,k}, b_j$ for each propositional variable p_j in ψ, one individual name c_i for each clause C_i, one individual name f for the whole formula ψ, two individual names $true, a_{true}$ for the propositional constant $true$, two individual names $false, a_{false}$ for the propositional constant $false$, and one individual name d for the marking mechanism which deals with the case that occurs whenever an individual corresponding to a propositional variable or constant belongs to both $\exists F^{k+1}.\top$ and $\exists F.(A \sqcap B)$.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{&Cl(f, c_1), Cl(f, c_2), \ldots, Cl(f, c_n), Cl(f, d),\\
&P_1(c_1, q_{1,1+}), P_2(c_1, q_{1,2+}), N_1(c_1, q_{1,1-}), N_2(c_1, q_{1,2-}),\\
&\ldots,\\
&P_1(c_n, q_{n,1+}), P_2(c_n, q_{n,2+}), N_1(c_n, q_{n,1-}), N_2(c_n, q_{n,2-}),\\
&P_1(d, p_1), P_2(d, p_1), N_1(d, p_1), N_2(d, p_1),\\
&\ldots,\\
&P_1(d, p_m), P_2(d, p_m), N_1(d, p_m), N_2(d, p_m),\\
&P_1(d, true), P_2(d, true), N_1(d, true), N_2(d, true),\\
&P_1(d, false), P_2(d, false), N_1(d, false), N_2(d, false),\\
&F(p_1, a_{1,1}), \ldots, F(p_1, a_{1,k}), F(p_1, b_1),\\
&\ldots,\\
&F(p_m, a_{m,1}), \ldots, F(p_1, a_{m,k}), F(p_1, b_m),\\
&F(a_{1,1}, a_{1,2}), F(a_{1,1}, a_{1,3}), \ldots, F(a_{1,1}, a_{1,k}),\\
&F(a_{1,2}, a_{1,3}), \ldots, F(a_{1,2}, a_{1,k}),\\
&\ldots,\\
&F(a_{1,k-1}, a_{1,k}),\\
&\ldots,
\end{aligned}$$

$$
\begin{aligned}
&F(a_{m,1}, a_{m,2}), F(a_{m,1}, a_{m,3}), \dots, F(a_{m,1}, a_{m,k}),\\
&F(a_{m,2}, a_{m,3}), \dots, F(a_{m,2}, a_{m,k}),\\
&\dots,\\
&F(a_{m,k-1}, a_{m,k}),\\
&A(a_{1,1}), \dots, A(a_{1,k}), B(b_1),\\
&\dots,\\
&A(a_{m,1}), \dots, A(a_{m,k}), B(b_m),\\
&F(true, a_{true}), F(a_{true}, a_{true}), F(false, a_{false}), A(a_{false}), B(a_{false})\},\\
Q \coloneqq\ &\exists Cl.(\exists P_1 \exists F.(A \sqcap B) \sqcap \exists P_2 \exists F.(A \sqcap B) \sqcap \exists N_1 \exists F^{k+1}.\top \sqcap \exists N_2 \exists F^{k+1}.\top)
\end{aligned}
$$

where for all $i = 1, \dots, n$, $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \dots, p_m, true, false\}$.

Claim: ψ is unsatisfiable if and only if $(\emptyset, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: First, we deal with the "only if" direction. Suppose ψ is unsatisfiable. Let $\mathcal{J}$ be a model of $\mathcal{A}_\psi$. We first show that

$$\{p_1^{\mathcal{J}}, \dots, p_m^{\mathcal{J}}, true^{\mathcal{J}}, false^{\mathcal{J}}\} \subseteq (\exists F^{k+1}.\top)^{\mathcal{J}} \cup (\exists F.(A \sqcap B))^{\mathcal{J}}$$

Since $\mathcal{J}$ is a model of $\mathcal{A}_\psi$, $true^{\mathcal{J}} \in (\exists F^{k+1}.\top)^{\mathcal{J}}$, and $false^{\mathcal{J}} \in (\exists F.(A \sqcap B))^{\mathcal{J}}$. In addition, for each $i = 1, \dots, m$, we have $(p_i^{\mathcal{J}}, a_{i,1}^{\mathcal{J}}) \in F^{\mathcal{J}}, \dots, (p_i^{\mathcal{J}}, a_{i,k}^{\mathcal{J}}) \in F^{\mathcal{J}}$, and $(p_i^{\mathcal{J}}, b_i^{\mathcal{J}}) \in F^{\mathcal{J}}$. Since F is globally k-functional, for each $i = 1, \dots, m$, there are two individual names, say x and y, from $a_{i,1}, \dots, a_{i,k}, b_i$ which must be identified. We now distinguish two cases:

- None of x, y is b_i. Then for some j_1, j_2, $1 \leq j_1 < j_2 \leq k$, a_{i,j_1} and a_{i,j_2} are identified. Since $(a_{i,j_1}^{\mathcal{J}}, a_{i,j_2}^{\mathcal{J}}) \in F^{\mathcal{J}}$ holds, we have that the individual $a_{i,j_1}^{\mathcal{J}} = a_{i,j_2}^{\mathcal{J}}$ is its own F-filler. In this case, it is easy to see that $p_i^{\mathcal{J}} \in (\exists F^{k+1}.\top)^{\mathcal{J}}$.
- One of x, y is b_i. Then for some j, $1 \leq j \leq k$, $a_{i,j}$ and b_i are identified, i.e., $a_{i,j}^{\mathcal{J}} = b_i^{\mathcal{J}}$. Since $a_{i,j}^{\mathcal{J}} \in A^{\mathcal{J}}$ and $b_i^{\mathcal{J}} \in B^{\mathcal{J}}$, we obtain $a_{i,j}^{\mathcal{J}} = b_i^{\mathcal{J}} \in (A \sqcap B)^{\mathcal{J}}$. In this case, it is obvious that $p_i \in (\exists F.(A \sqcap B))^{\mathcal{J}}$.

Overall, we conclude that $\{p_1^{\mathcal{J}}, \dots, p_m^{\mathcal{J}}, true^{\mathcal{J}}, false^{\mathcal{J}}\} \subseteq (\exists F^{k+1}.\top)^{\mathcal{J}} \cup (\exists F.(A \sqcap B))^{\mathcal{J}}$ indeed holds.

Next, we show that $f^{\mathcal{J}} \in Q^{\mathcal{J}}$. Note that $(\exists F^{k+1}.\top)^{\mathcal{J}} \cap (\exists F.(A \sqcap B))^{\mathcal{J}}$ is not necessarily empty. If there is a $q \in \{p_1, \dots, p_m, true, false\}$ with $q^{\mathcal{J}} \in (\exists F^{k+1}.\top)^{\mathcal{J}} \cap (\exists F.(A \sqcap B))^{\mathcal{J}}$, then $f^{\mathcal{J}} \in (\exists Cl.(\exists P_1 \exists F.(A \sqcap B) \sqcap \exists P_2 \exists F.(A \sqcap B) \sqcap \exists N_1 \exists F^{k+1}.\top \sqcap \exists N_2 \exists F^{k+1}.\top))^{\mathcal{J}} = Q^{\mathcal{J}}$ is due to the fact that $d^{\mathcal{J}}$ is a Cl-filler of $f^{\mathcal{J}}$, and $q^{\mathcal{J}}$ is simultaneously a P_1-, P_2-, N_1-, and N_2-filler of $d^{\mathcal{J}}$. Otherwise, for all $q \in \{p_1, \dots, p_m, true, false\}$, we have either $q \in (\exists F^{k+1}.\top)^{\mathcal{J}}$ or $q \in (\exists F.(A \sqcap B))^{\mathcal{J}}$ but not both. In this case, like in the $\mathcal{EL}^{\forall}$ case on page 25, it is not difficult to show that unsatisfiability of ψ implies $f^{\mathcal{J}} \in Q^{\mathcal{J}}$ via one of $c_i^{\mathcal{J}}$, $i = 1, \dots, n$.

For the "if" direction, assume that ψ is satisfiable and let δ be a truth assignment that satisfies ψ. We show that $(\emptyset, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model of $\mathcal{A}_\psi$ that satisfies $Q(f)$. We construct an interpretation $\mathcal{I}_\delta$ below.

- $\Delta^{\mathcal{I}_\delta} = \{f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false, d, a_{true}, a_{false}, t_1, \ldots, t_k, f_1, \ldots, f_k\}$
- $x^{\mathcal{I}_\delta} = x$ for all $x \in \{f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false, d, a_{true}, a_{false}\}$
- for every $i = 1, \ldots, m$, $j = 1, \ldots, k-1$, $a_{i,j}^{\mathcal{I}_\delta} = \begin{cases} t_j, & \text{if } \delta(p_i) = 1, \\ f_j, & \text{if } \delta(p_i) = 0 \end{cases}$
- for every $i = 1, \ldots, m$, $a_{i,k}^{\mathcal{I}_\delta} = \begin{cases} t_{k-1}, & \text{if } \delta(p_i) = 1, \\ f_k, & \text{if } \delta(p_i) = 0 \end{cases}$
- for every $i = 1, \ldots, m$, $b_i^{\mathcal{I}_\delta} = \begin{cases} t_k, & \text{if } \delta(p_i) = 1, \\ f_k, & \text{if } \delta(p_i) = 0 \end{cases}$
- $A^{\mathcal{I}_\delta} = \{a_{false}, t_1, \ldots, t_{k-1}, f_1, \ldots, f_k\}$
- $B^{\mathcal{I}_\delta} = \{a_{false}, t_k, f_k\}$
- $F^{\mathcal{I}_\delta} = F_0 \cup F_1 \cup F_2 \cup \{(true, a_{true}), (a_{true}, a_{true}), (false, a_{false})\}$, where

$$F_0 = \bigcup_{j=1}^{k} \{(p^{\mathcal{I}_\delta}, f_j) \mid \delta(p) = 0, p \in \{p_1, \ldots, p_m\}\}$$
$$F_1 = \bigcup_{j=1}^{k} \{(p^{\mathcal{I}_\delta}, t_j) \mid \delta(p) = 1, p \in \{p_1, \ldots, p_m\}\}$$
$$F_2 = \{(t_{k-1}, t_{k-1})\} \cup \{(t_{j_1}, t_{j_2}) \mid 1 \leq j_1 < j_2 \leq k-1\} \cup \{(f_{j_1}, f_{j_2}) \mid 1 \leq j_1 < j_2 \leq k\}$$

- $\rho^{\mathcal{I}_\delta} = \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \mid \rho(x, y) \in \mathcal{A}_\psi\}$, for $\rho \in \{Cl, P_1, P_2, N_1, N_2\}$

It is straightforward to verify that $\mathcal{I}_\delta$ is indeed a model of $(\mathcal{T}, \mathcal{A}_\psi)$. Moreover, it is not hard to show that satisfiability of ψ implies $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$ (see the proof of the claim that leads to Proposition 3.8 on page 25). $\diamond$

Hence, together with the fact that the size of the ABox $\mathcal{A}_\psi$ is polynomial in the size of the 2+2-CNF formula ψ, the claim yields the coNP-hardness regarding data complexity of instance checking in $\mathcal{EL}^{kf}$ w.r.t. simple ABoxes and without TBoxes. In addition, since data complexity of instance checking w.r.t. simple ABoxes for $\mathcal{SHIQ}$, a superlanguage of $\mathcal{EL}^{kf}$, is in coNP, we establish the following coNP-completeness result which also holds for acyclic and general TBoxes.

Proposition 3.11
*Data complexity of instance checking in $\mathcal{EL}^{kf}$ w.r.t. simple ABoxes and without TBoxes is co*NP*-complete.* $\diamond$

3.4.2 Adding Only At-Least Restriction $(\geq k\ r)$: $\mathcal{EL}^{\geq k}$

Next, we consider locally adding at-least restriction $(\geq k\ r)$ to $\mathcal{EL}$. The resulting DLs are named $\mathcal{EL}^{\geq k}$. Obviously, it only makes sense to consider $k > 1$, because $(\geq 0\ r)$ is trivially equivalent to $\top$ and for $k = 1$, the resulting DL is just $\mathcal{EL}$. For $k > 1$, it turns out that we need a nonempty TBox as part of the input of instance checking to establish coNP-hardness regarding data complexity, because without a TBox, data complexity of instance checking is polynomial which will also be proved at the end of this subsection. For the coNP upper bound, like in the previous sections, we refer again to the result from [Hustadt *et al.*, 2005] which showed that data complexity of instance checking w.r.t. simple ABoxes for $\mathcal{SHIQ}$, a superlanguage of $\mathcal{EL}^{\geq k}$ is coNP-complete, regardless of which TBox formalism that is used.

Instance Checking for $\mathcal{EL}^{\geq k}$ w.r.t. Acyclic TBoxes

If we require the TBox in the knowledge base input of instance checking to be acyclic, coNP-hardness regarding data complexity is obtained only for $\mathcal{EL}^{\geq 2}$, i.e., $\mathcal{EL}$ extended with $(\geq 2\ r)$. For $k > 2$, we left it as an open problem.

In the following, we show that data complexity of instance checking in $\mathcal{EL}^{\geq 2}$ w.r.t. acyclic TBoxes is coNP-hard. To establish this lower bound, we employ a reduction from the complement of 2+2-SAT similar to the one for $\mathcal{EL}^{\forall}$. Here, we need to simulate the way $\exists S.\top$ and $\forall S.A$ behave in the $\mathcal{EL}^{\forall}$ case. This can be accomplished by using the pair of concepts $(\geq 2\ S)$ and $\exists S.(A \sqcap B)$. The union of this pair is equivalent to the top-concept if $\top \equiv \exists S.A \sqcap \exists S.B$ is satisfied. But $\top \equiv \exists s.A \sqcap \exists s.B$ cannot be expressed by using acyclic TBoxes. Fortunately, as already seen in the $\mathcal{ELU}$ case, it suffices to consider only individuals that correspond to propositional variables and constants. Hence, we can use the same trick used in the $\mathcal{ELU}$ case (see Section 3.2) in which we use another concept name A_{var}, instead of the top-concept. Here, A_{var} is used in a concept assertion for every individual in the ABox that is associated with a propositional variable or constant. The acyclic TBox is then obtained with the following three concept definitions:

$$A_{var} \equiv \exists S.A \sqcap \exists S.B \qquad A_T \equiv A_{var} \sqcap \exists S.(A \sqcap B) \qquad A_F \equiv A_{var} \sqcap (\geq 2\ S) \tag{3.7}$$

These concept definitions ensure that the union of A_T and A_F is equivalent to A_{var}. Now it remains to deal with the case in which there is an instance of A_{var} that is an instance of both A_T and A_F. We solve this using an additional individual in the ABox that performs as a marker for every individual that belongs to $A_T \sqcap A_F$. This marking mechanism is the same as the one introduced in the $\mathcal{EL}^{\forall}$ case (see Subsection 3.3.2, page 24).

Let $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ be a 2+2-CNF formula with m propositional variables and n clauses where each clause is of the form $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. We use the same individuals as in $\mathcal{ELU}$ ABox given in (3.2), but with an additional individual d. The simple $\mathcal{EL}^{\geq 2}$ ABox $\mathcal{A}_\psi$ then has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, and two individuals $true$ and $false$ for the corresponding propositional constants.

Additionally, it also contains one individual d. In this reduction, the following concept names are used: A_{var}, A_T, A_F, B_1, B_2; and the following role names are used: Cl, P_1, $P2$, N_1, N_2, S. The simple ABox $\mathcal{A}_\psi$, the acyclic $\mathcal{EL}^{\geq 2}$ TBox $\mathcal{T}$ and the $\mathcal{EL}^{\geq 2}$ query concept Q are described below.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{&A_T(true), A_F(false),\\
&Cl(f,c_1), Cl(f,c_2), \ldots, Cl(f,c_n), Cl(f,d),\\
&P_1(c_1,q_{1,1+}), P_2(c_1,q_{1,2+}), N_1(c_1,q_{1,1-}), N_2(c_1,q_{1,2-}),\\
&\ldots,\\
&P_1(c_n,q_{n,1+}), P_2(c_n,q_{n,2+}), N_1(c_n,q_{n,1-}), N_2(c_n,q_{n,2-}),\\
&P_1(d,p_1), P_2(d,p_1), N_1(d,p_1), N_2(d,p_1),\\
&\ldots, \qquad\qquad (3.8)\\
&P_1(d,p_m), P_2(d,p_m), N_1(d,p_m), N_2(d,p_m),\\
&P_1(d,true), P_2(d,true), N_1(d,true), N_2(d,true),\\
&P_1(d,false), P_2(d,false), N_1(d,false), N_2(d,false),\\
&A_{var}(p_1), A_{var}(p_2), \ldots, A_{var}(p_m)\},\\
\mathcal{T} \coloneqq \{&A_{var} \equiv \exists S.B_1 \sqcap \exists S.B_2,\\
&A_T \quad \equiv A_{var} \sqcap (\geq 2\ S),\\
&A_F \quad \equiv A_{var} \sqcap \exists S.(B_1 \sqcap B_2)\},\\
Q \coloneqq\ &\exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)
\end{aligned}$$

where for all $i = 1, \ldots, n$, $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \ldots, p_m, true, false\}$.

Claim: ψ is unsatisfiable if and only if $(\mathcal{T}, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: The "only if" part is very similar to the one in the $\mathcal{EL}^\forall$ case (the proof of the claim on page 25), only this time, we use an observation that $A_{var}^{\mathcal{I}} = A_T^{\mathcal{I}} \cup A_F^{\mathcal{I}}$ for every model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A}_\psi)$.

For the "if" part, suppose ψ is satisfiable. We show that $(\mathcal{T}, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model of $\mathcal{T}$ and $\mathcal{A}_\psi$ that does not satisfy $Q(f)$. Let δ be a truth assignment that satisfies ψ. We define an interpretation $\mathcal{I}_\delta$ for $\mathcal{T}, \mathcal{A}_\psi$ and Q as follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_\delta} &= \{d, e_0, e_1, e_2, f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\}\\
x^{\mathcal{I}_\delta} &= x, \text{ for all } x \in \{d, f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\}\\
A_{var}^{\mathcal{I}_\delta} &= \{p_1, \ldots, p_m, true, false\}\\
A_T^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 1, p \in \{true, p_1, \ldots, p_m\}\}\\
A_F^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 0, p \in \{false, p_1, \ldots, p_m\}\}\\
B_1^{\mathcal{I}_\delta} &= \{e_0, e_1\},\ B_2^{\mathcal{I}_\delta} = \{e_0, e_2\}\\
S^{\mathcal{I}_\delta} &= \{(p^{\mathcal{I}_\delta}, e_0) \mid \delta(p) = 0, p \in \{false, p_1, \ldots, p_m\}\}\\
&\cup \{(p^{\mathcal{I}_\delta}, e_1) \mid \delta(p) = 1, p \in \{true, p_1, \ldots, p_m\}\}
\end{aligned}$$

$$\cup \{(p^{\mathcal{I}_\delta}, e_2) \mid \delta(p) = 1, p \in \{true, p_1, \ldots, p_m\}\}$$
$$\rho^{\mathcal{I}_\delta} = \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \mid \rho(x,y) \in \mathcal{A}_\psi\} \quad \text{for } \rho \in \{Cl, P_1, P_2, N_1, N_2\}$$

It is straightforward to verify that $\mathcal{I}_\delta$ is a model of $(\mathcal{T}, \mathcal{A}_\psi)$. Moreover, it is easy to show that satisfiability of ψ implies $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$ (see the proof of the claim on page 25 which yields Proposition 3.8). $\diamond$

Since the size of the ABox $\mathcal{A}_\psi$ is polynomial in the size of the 2+2-CNF formula ψ, the previous claim yields the coNP-hardness regarding data complexity of instance checking w.r.t. simple ABoxes and acyclic TBoxes in $\mathcal{EL}^{\geq 2}$. Hence, together with the fact that data complexity of $\mathcal{SHIQ}$, a superlanguage of $\mathcal{EL}^{\geq 2}$, w.r.t. simple ABoxes is in coNP, we obtain the following proposition.

Proposition 3.12
*Data complexity of instance checking in $\mathcal{EL}^{\geq 2}$ w.r.t. simple ABoxes and acyclic TBoxes is co*NP*-complete.* $\diamond$

Instance Checking for $\mathcal{EL}^{\geq k}$ w.r.t. General TBoxes

In view of Proposition 3.12, a natural question is whether analogous results can be obtained for $\mathcal{EL}^{\geq k}$ with $k > 2$. Apparently, this can be done if we allow GCIs in the TBox.

We first informally describe the idea for the reduction, e.g., for $k = 3$. The idea of the reduction follows from an easily proved fact that a TBox containing the following GCIs:

$$\begin{aligned}
&A \sqsubseteq \exists S.B_1 \sqcap \exists S.B_2 \sqcap \exists S.B_3,\\
&A \sqcap (\geq 3\ S) \sqsubseteq A_0,\\
&A \sqcap \exists S.(B_1 \sqcap B_2) \sqsubseteq A_1,\\
&A \sqcap \exists S.(B_1 \sqcap B_3) \sqsubseteq A_2,\\
&A \sqcap \exists S.(B_2 \sqcap B_3) \sqsubseteq A_3,
\end{aligned}$$

simulates the GCI $A \sqsubseteq A_0 \sqcup A_1 \sqcup A_2 \sqcup A_3$. Hence, whenever an individual name belongs to A, a case analysis is required to determine whether c belongs to A_0 A_1, A_2 or A_3.

For establishing a coNP-hardness result, the aforementioned case analysis ought to be related to the truth assignments of a 2+2-CNF formula. The concept name A in the above explanation performs as the concept name $\mathcal{A}_{var}$, the concept name that is true precisely at every individual name which is associated to either a propositional variable or constant. Two among these four concept names: A_0 A_1, A_2 and A_3 perform as A_T and A_F, the pair of concepts associated to the truth values 1 and 0. The remaining concept name performs as dummy concept names $A_D, A_{D'}$.

Consider an individual name c that is associated to a propositional variable or constant. We enforce c to belong to A_{var} by setting $A_{var}(c)$ in the ABox. Since A_{var} is subsumed by the union of A_T, A_F, A_D, and $A_{D'}$, c must belong to at least one of those four concept names. In this situation, we need to consider three cases of which two are "bad". The "good" case is whenever c belongs to either A_T or A_F but not both. In this case, the truth value of the propositional variable or constant associated to c precisely corresponds to whether c belongs to A_T or A_F. The other two "bad" cases are, first, whenever c belongs to both A_T and A_F, and second, whenever c belongs to neither A_T nor A_F. The first case is solved by employing a new individual name d for a marking mechanism which is also used in the reduction for $\mathcal{EL}^{\forall}$ (see subsection 3.3.2). For the second case, we use another individual name a for a second marking mechanism that works as follows. The individual name a belongs to a concept D and is connected through a role R to every individual which is associated to either a propositional variable or constant. Whenever c belongs to neither A_T nor A_F, then it must belong to either A_D or $A_{D'}$. In this case, we force a to belong to both A_T and A_F using two GCIs:

$$D \sqcap \exists R.A_D \sqsubseteq A_T \sqcap A_F$$
$$D \sqcap \exists R.A_{D'} \sqsubseteq A_T \sqcap A_F$$

Finally, this marking mechanism is completed as now a belongs to both A_T and A_F, and this can be dealt with using the first marking mechanism.

We describe here a reduction from the complement of 2+2-SAT to instance checking in $\mathcal{EL}^{\geq k}$ with $k > 2$. To illustrate how it is done, we first show an example of such reduction for $k = 3$. We start with a 2+2-CNF formula $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$ with m propositional variables and n clauses where each clause is of the form $C_i = q_{i,1+} \vee q_{i,2+} \vee \neg q_{i,1-} \vee \neg q_{i,2-}$. The $\mathcal{EL}^{\geq 3}$ ABox $\mathcal{A}_\psi$ has one individual p_j for each propositional variable p_j in ψ, one individual c_i for each clause C_i, one individual f for the whole formula ψ, and two individuals $true$ and $false$ corresponding to propositional constants. Additionally, $\mathcal{A}_\psi$ also has three other individuals: a, b and d. In this reduction, the following concept names are used: A_{var}, A_T, A_F, A_D, $A_{D'}$, B_1, B_2, B_3, D; and the following role names are used: Cl, P_1, $P2$, N_1, N_2, R, S. The ABox $\mathcal{A}_\psi$, the TBox $\mathcal{T}$ and the query concept Q are described below.

$$\begin{aligned}
\mathcal{A}_\psi \coloneqq \{&A_T(true), A_F(false),\\
&Cl(f,c_1), Cl(f,c_2), \ldots, Cl(f,c_n), Cl(f,d),\\
&P_1(c_1,q_{1,1+}), P_2(c_1,q_{1,2+}), N_1(c_1,q_{1,1-}), N_2(c_1,q_{1,2-}),\\
&\ldots,\\
&P_1(c_n,q_{n,1+}), P_2(c_n,q_{n,2+}), N_1(c_n,q_{n,1-}), N_2(c_n,q_{n,2-}),\\
&P_1(d,p_1), P_2(d,p_1), N_1(d,p_1), N_2(d,p_1),\\
&\ldots,\\
&P_1(d,p_m), P_2(d,p_m), N_1(d,p_m), N_2(d,p_m),\\
&P_1(d,true), P_2(d,true), N_1(d,true), N_2(d,true),\\
&P_1(d,false), P_2(d,false), N_1(d,false), N_2(d,false),\\
&P_1(d,a), P_2(d,a), N_1(d,a), N_2(d,a),\\
&A_{var}(p_1), A_{var}(p_2), \ldots, A_{var}(p_m),\\
&D(a), R(a,p_1), \ldots, R(a,p_m)\},
\end{aligned}$$

$$\begin{aligned}
\mathcal{T} \coloneqq \{&A_{var} \sqsubseteq \exists S.B_1 \sqcap \exists S.B_2 \sqcap \exists S.B_3, && \text{(3.9a)}\\
&A_{var} \sqcap (\geq 3\ S) \sqsubseteq A_T && \text{(3.9b)}\\
&A_{var} \sqcap \exists S.(B_1 \sqcap B_2) \sqsubseteq A_F, && \text{(3.9c)}\\
&A_{var} \sqcap \exists S.(B_1 \sqcap B_3) \sqsubseteq A_D, && \text{(3.9d)}\\
&A_{var} \sqcap \exists S.(B_2 \sqcap B_3) \sqsubseteq A_{D'}, && \text{(3.9e)}\\
&D \sqcap \exists R.A_D \sqsubseteq A_T \sqcap A_F, && \text{(3.9f)}\\
&D \sqcap \exists R.A_{D'} \sqsubseteq A_T \sqcap A_F\}, && \text{(3.9g)}\\
Q \coloneqq\ &\exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)
\end{aligned}$$

where for all $i = 1, \ldots, n$, $q_{i,1+}, q_{i,2+}, q_{i,1-}, q_{i,2-} \in \{p_1, p_2, \ldots, p_m, true, false\}$.

Claim: ψ is unsatisfiable if and only if $(\mathcal{T}, \mathcal{A}_\psi) \models Q(f)$.

Proof of Claim: First, we deal with the "only if" direction. Suppose ψ is unsatisfiable. Assume that $\mathcal{J}$ is a model of $\mathcal{T}$ and $\mathcal{A}_\psi$. We first show that $A_{var}^{\mathcal{J}} \subseteq A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}} \cup A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$. Suppose $x \in A_{var}^{\mathcal{J}}$. Since $\mathcal{J}$ is a model of $\mathcal{T}$, x has one S-filler that belongs to $B_1^{\mathcal{J}}$, one S-filler that belongs to $B_2^{\mathcal{J}}$ and one S-filler that belongs to $B_3^{\mathcal{J}}$. Consider the set $M = \{y \mid (x, y) \in S^{\mathcal{J}}\}$ of all S-fillers of x. Since $\mathcal{J}$ is a model of $\mathcal{T}$, it is clear that $x \in (\exists S.B_1 \sqcap \exists S.B_2 \sqcap \exists S.B_3)^{\mathcal{J}}$, i.e., $M \neq \emptyset$. If $|M| \geq 3$ then $x \in A_T^{\mathcal{J}} \subseteq A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}} \cup A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$. Otherwise, there is an S-filler of x, say y, such that either $y \in B_1^{\mathcal{J}} \cap B_2^{\mathcal{J}}$ or $y \in B_1^{\mathcal{J}} \cap B_3^{\mathcal{J}}$ or $y \in B_2^{\mathcal{J}} \cap B_3^{\mathcal{J}}$. Hence, either $x \in A_F^{\mathcal{J}}$ or $x \in A_D^{\mathcal{J}}$ or $x \in A_{D'}^{\mathcal{J}}$. This again implies $x \in A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}} \cup A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$ as required.

Now we show that $f^{\mathcal{J}} \in Q^{\mathcal{J}}$. Since for all $p^{\mathcal{J}} \in \{p_1^{\mathcal{J}}, \ldots, p_m^{\mathcal{J}}\}$, $p^{\mathcal{J}} \in A_{var}^{\mathcal{J}}$, we have $p^{\mathcal{J}} \in A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}} \cup A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$. Moreover, we also have $\{true^{\mathcal{J}}, false^{\mathcal{J}}\} \subseteq A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}} \cup A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$. We now distinguish three cases as follows.

- There is an $q \in \{true, false, p_1, \ldots, p_m\}$ such that $q^{\mathcal{J}} \notin A_T^{\mathcal{J}} \cup A_F^{\mathcal{J}}$. Hence, $q^{\mathcal{J}} \in A_D^{\mathcal{J}} \cup A_{D'}^{\mathcal{J}}$. If $q^{\mathcal{J}} \in A_D^{\mathcal{J}}$ then obviously $a^{\mathcal{J}} \in (A_T \sqcap A_F)^{\mathcal{J}}$. Similarly, $q^{\mathcal{J}} \in A_{D'}^{\mathcal{J}}$ also implies $a^{\mathcal{J}} \in (A_T \sqcap A_F)^{\mathcal{J}}$. From either case, $d^{\mathcal{J}} \in (\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)^{\mathcal{J}}$. We thus conclude $f^{\mathcal{J}} \in (\exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T))^{\mathcal{J}}$, i.e., $f^{\mathcal{J}} \in Q^{\mathcal{J}}$.

- There is a $q \in \{true, false, p_1, \ldots, p_m\}$ such that $q^{\mathcal{J}} \in A_T^{\mathcal{J}} \cap A_F^{\mathcal{J}}$. Then we obtain that $d^{\mathcal{J}} \in (\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T)^{\mathcal{J}}$. Thus, $f^{\mathcal{J}} \in (\exists Cl.(\exists P_1.A_F \sqcap \exists P_2.A_F \sqcap \exists N_1.A_T \sqcap \exists N_2.A_T))^{\mathcal{J}}$, i.e., $f^{\mathcal{J}} \in Q^{\mathcal{J}}$.

- For all $q \in \{true, false, p_1, \ldots, p_m\}$, either $q \in A_T^{\mathcal{J}}$ or $q \in A_F^{\mathcal{J}}$ but not both. In this case, like in the $\mathcal{EL}^{\forall}$ case on page 25, it is not difficult to show that unsatisfiability of ψ implies $f^{\mathcal{J}} \in Q^{\mathcal{J}}$ via one of $c_i^{\mathcal{J}}$, $i = 1, \ldots, n$.

For the "if" direction, assume that ψ is satisfiable and let δ be a truth assignment that satisfies ψ. We show that $(\mathcal{T}, \mathcal{A}_\psi) \not\models Q(f)$, i.e., there is a model $\mathcal{I}_\delta$ of $(\mathcal{T}, \mathcal{A}_\psi)$ such that $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$. Such model $\mathcal{I}_\delta$ of $(\mathcal{T}, \mathcal{A}_\psi)$ is constructed below.

$$\Delta^{\mathcal{I}_\delta} = \{e_0, e_1, e_2, e_3, a, d, f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\}$$

$$
\begin{aligned}
x^{\mathcal{I}_\delta} &= x, \text{ for all } x \in \{a, d, f, c_1, \ldots, c_n, p_1, \ldots, p_m, true, false\} \\
D^{\mathcal{I}_\delta} &= \{a\},\ A_D^{\mathcal{I}_\delta} = A_{D'}^{\mathcal{I}_\delta} = \emptyset \\
A_{var}^{\mathcal{I}_\delta} &= \{p_1, \ldots, p_m\} \\
B_1^{\mathcal{I}_\delta} &= \{e_1, e_3\},\ B_2^{\mathcal{I}_\delta} = \{e_2, e_3\},\ B_3^{\mathcal{I}_\delta} = \{e_0\} \\
A_T^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 1, p \in \{p_1, \ldots, p_m\}\} \\
A_F^{\mathcal{I}_\delta} &= \{p^{\mathcal{I}_\delta} \mid \delta(p) = 0, p \in \{p_1, \ldots, p_m\}\} \\
S^{\mathcal{I}_\delta} &= \{(p^{\mathcal{I}_\delta}, e_0) \mid p \in \{p_1, \ldots, p_m\}\} \\
&\qquad \cup \{(p^{\mathcal{I}_\delta}, e_1) \mid \delta(p) = true, p \in \{p_1, \ldots, p_m\}\} \\
&\qquad \cup \{(p^{\mathcal{I}_\delta}, e_2) \mid \delta(p) = true, p \in \{p_1, \ldots, p_m\}\} \\
&\qquad \cup \{(p^{\mathcal{I}_\delta}, e_3) \mid \delta(p) = 0, p \in \{p_1, \ldots, p_m\}\} \\
\rho^{\mathcal{I}_\delta} &= \{(x^{\mathcal{I}_\delta}, y^{\mathcal{I}_\delta}) \mid \rho(x, y) \in \mathcal{A}_\psi\} \quad \text{for } \rho \in \{Cl, P_1, P_2, N_1, N_2, R\}
\end{aligned}
$$

It is straightforward to verify that $\mathcal{I}_\delta$ is indeed a model of $(\mathcal{T}, \mathcal{A}_\psi)$. Moreover, it is not hard to show that satisfiability of ψ implies $f^{\mathcal{I}_\delta} \notin Q^{\mathcal{I}_\delta}$ (see the proof of the claim that leads to Proposition 3.8 on page 25). $\diamond$

Since the size of the ABox $\mathcal{A}_\psi$ is polynomial in the size of the 2+2-CNF formula ψ, we obtain coNP-hardness regarding data complexity for instance checking in $\mathcal{EL}^{\geq 3}$ w.r.t. general TBoxes. Moreover, as data complexity of instance checking w.r.t. simple ABoxes in $\mathcal{SHIQ}$, a superlanguage of $\mathcal{EL}^{\geq 3}$, is in coNP [Hustadt *et al.*, 2005], we have the following proposition.

Proposition 3.13
*Data complexity of instance checking in $\mathcal{EL}^{\geq 3}$ w.r.t. simple ABoxes and general TBoxes is co*NP*-complete.* $\diamond$

An analogous result as the previous proposition for $\mathcal{EL}^{\geq k}$ with $k > 3$, can be derived in a similar way as above. We only need to modify the TBox for the reduction. In the TBox, we replace the GCI in (3.9a) with a GCI of the form $A_{var} \sqsubseteq \exists S.B_1 \sqcap \ldots \sqcap \exists S.B_k$ which imposes any individual that belongs to A_{var} to have at least one S-filler. Consider an individual which belongs to A_{var}. First, it can have at least k S-fillers. Thus, we replace the GCI in (3.9b) with a GCI of the form $A_{var} \sqcap (\geq k\ S) \sqsubseteq A_T$. If it has less than k S-filler than one of these S-fillers belongs simultaneously to two concepts among $B_1, \ldots, B_k$. There are $\binom{k}{2}$ possible pairs formed from these k concepts. Thus, we replace all GCIs (3.9c)–(3.9e) with GCIs of the form $A_{var} \sqcap \exists S.(B_i \sqcap B_j) \sqsubseteq A$ where $1 \leq i < j \leq k$ and $A \in \{A_F, A_{D_1}, \ldots, A_{D_{k'}}\}$, $k' = \binom{k}{2} - 1$. Here, the concept names A_{D_n}, $1 \leq n \leq \binom{k}{2} - 1$ perform as dummy concept names.

For the reduction, we would like to have a "good" truth assignment only if every individual name that is associated to a propositional variable or constant belongs to A_T or A_F and not both. Thus, whenever such an individual belongs to one of these dummy concept names, we view it as a "bad" truth assignment. Since the query concept only uses two concept names A_T and A_F, in order to detect such a bad truth assignment, there has to be an individual which is an instance of both A_T and A_F. Thus, we use an individual a as a marker such that whenever there is an individual belonging to one of the dummy concept names, a belongs to both A_T and A_F. This is done by replacing GCIs (3.9f)–(3.9g) with $\binom{k}{2} - 1$ GCIs of the form $D \sqcap \exists R.A_{D_n} \sqsubseteq A_T \sqcap A_F$, $1 \leq n \leq \binom{k}{2} - 1$. To complete the reduction, we also need to have another individual d which handles the marking mechanism for detecting any individuals that belong to both A_T and A_F.

Overall, the size of TBox in the reduction is quadratic in k. But this does not matter since k does not depend on the size of 2+2-CNF formula ψ. Note that the size of the ABox $\mathcal{A}_\psi$ is polynomial in the size of ψ. Hence, together with the coNP upper bound from the result of [Hustadt *et al.*, 2005], we conclude with the following proposition.

Proposition 3.14
*For fixed nonnegative integers $k \geq 3$, data complexity of instance checking in $\mathcal{EL}^{\geq k}$ w.r.t. simple ABoxes and general TBoxes is co*NP*-complete.* $\diamond$

Instance Checking for $\mathcal{EL}^{\geq k}$ without TBoxes

As already mentioned at the beginning of the current subsection, we now proceed with the investigation on data complexity of instance checking for $\mathcal{EL}^{\geq k}$ without TBoxes. Unlike $\mathcal{EL}^{\leq k}$ for which data complexity of instance checking is already coNP-complete, even without the presence of a TBox, data complexity of instance checking in $\mathcal{EL}^{\geq k}$ is polynomial if we disallow a TBox as part of the input. Note that in $\mathcal{EL}^{\geq k}$, for a simple ABox $\mathcal{A}$, a role name r, and an individual name d, an at-least restriction $(\geq k\ r)$ is true at d iff the number of r-fillers of d which are explicitly asserted in $\mathcal{A}$ is at least k, i.e.,

$$\mathcal{A} \models (\geq k\ r)(d) \text{ iff } |\{r(d, d') \in \mathcal{A}\}| \geq k$$

Thus, we can decide the instance checking in $\mathcal{EL}^{\geq k}$ without TBoxes using an algorithm that is similar to the one used for deciding instance checking in $\mathcal{ELU}$ without TBoxes which is described on page 21. In fact, we only need to replace the rule **(3)** there (which is the rule for handling disjunction) with the following rule where D is the $\mathcal{EL}^{\geq k}$ query concept.

(3) Let d be an individual name occurring in $\widehat{\mathcal{A}}$.
If $(\geq k\ r) \in \mathsf{sub}(D)$, $|\{r(d, d') \in \widehat{\mathcal{A}}\}| \geq k$, and $(\geq k\ r)(d) \notin \widehat{\mathcal{A}}$,
then $\widehat{\mathcal{A}} := \widehat{\mathcal{A}} \cup \{(\geq k\ r)(d)\}$;

Similar to Lemma 3.5, it is easy to show that the resulting algorithm is sound and complete, and runs in time polynomial in the size of the input ABox. Thus, we obtain the following proposition

Proposition 3.15
For all nonnegative integers k, data complexity of instance checking in $\mathcal{EL}^{\geq k}$ with simple ABoxes and without TBoxes is polynomial. $\diamond$

3.5 Extensions of $\mathcal{EL}$ with Role Complement, Role Union and Transitive Closures

For this section, we consider $\mathcal{EL}^{R\neg}$, $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^{*}$ which are respectively extensions of $\mathcal{EL}$ with role complement, role union and transitive closures. Another extension of $\mathcal{EL}$ with inverse roles (called $\mathcal{ELI}$) is not discussed here since data complexity of instance checking w.r.t. general TBoxes for $\mathcal{ELI}$ can be shown to be polynomial (see Chapter 4).

For $\mathcal{EL}^{R\neg}$, data complexity of instance checking is coNP-hard even without the presence of a TBox. This can be shown by a reduction from the complement of 2+2-SAT which is similar to the one for $\mathcal{EL}^{\forall}$. In fact, the reduction for the $\mathcal{EL}^{R\neg}$ translates a 2+2-CNF formula ψ with m variables and n clauses into the same ABox $\mathcal{A}_\psi$ in Equation (3.4) and the query concept $Q \coloneqq \exists Cl.(\exists P_1 \exists \neg S.\top \sqcap \exists P_2 \exists \neg S.\top \sqcap \exists N_1 \exists S.\top \sqcap \exists N_2 \exists S.\top)$. By observing the fact that for every interpretation $\mathcal{I}$, $\Delta^{\mathcal{I}} = (\exists S.\top)^{\mathcal{I}} \cup (\exists \neg S.\top)^{\mathcal{I}}$, it is easy to show that ψ is unsatisfiable iff $(\emptyset, \mathcal{A}_\psi) \models Q(f)$ where f is the individual name occurring in $\mathcal{A}_\psi$ that is associated by the reduction to the whole formula ψ. Thus, we obtain the following proposition.

Proposition 3.16
*Data complexity of instance checking in $\mathcal{EL}^{R\neg}$ with simple ABoxes and without TBoxes is co*NP*-hard.* ◇

Data complexity of instance checking w.r.t. acyclic TBoxes for $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^{*}$ can be proved coNP-hard using reductions that is similar to the one for $\mathcal{EL}^{\geq 2}$. For a 2+2-CNF formula ψ with m variables and n clauses, the reduction translates ψ into the same ABox A_ψ as in Equation (3.8). The query concept Q is also the same. The difference is only on the TBox $\mathcal{T}$ which is given below where $\mathcal{T}_1$ is for $\mathcal{EL}^{\cup}$ and $\mathcal{T}_2$ is for $\mathcal{EL}^{*}$, where R and S are new role names.

$$\mathcal{T}_1 \coloneqq \{A_{var} \equiv \exists R \cup S.\top, \quad A_T \equiv A_{var} \sqcap \exists R.\top, \quad A_F \equiv A_{var} \sqcap \exists S.\top\}; \tag{3.10}$$
$$\mathcal{T}_2 \coloneqq \{A_{var} \equiv \exists R^{+}.\top, \quad A_T \equiv A_{var} \sqcap \exists R.\top, \quad A_F \equiv A_{var} \sqcap \exists R \exists R^{+}.\top\} \tag{3.11}$$

Note that in Equations (3.10) and (3.11), the union of pairs of concepts A_T and A_F is equivalent to A_{var}. Hence, using a very similar reasoning to the one used for $\mathcal{EL}^{\geq 2}$, with $\mathcal{T}$ given by the previous two equations, the following claim holds:

Claim: ψ is unsatisfiable iff $(\mathcal{T}, \mathcal{A}_\psi) \models Q(f)$. ◇

Thus, analogous with Proposition 3.12, we obtain the following proposition.

Proposition 3.17
Data complexity of instance checking w.r.t. simple ABoxes and acyclic TBoxes for $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^{*}$ are coNP-hard. ◇

Now we are going to show that without TBoxes, data complexity of instance checking for $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^{*}$ are polynomial. The proofs are very similar to the one in the $\mathcal{ELU}$ case.

For $\mathcal{EL}^{\cup}$, an algorithm that decides instance checking with simple ABoxes and without TBoxes can be obtained by modifying the similar algorithm used for $\mathcal{ELU}$ on page 21.

More precisely, we replace the rule **(3)** which handles disjunction with the following rule where $\mathsf{rol}(D)$ is the set containing precisely every role name and complex roles occurring in the query concept D.

(3) If $r \sqcup s \in \mathsf{rol}(D)$, $\{r(d,d'), s(d,d')\} \cap \widehat{\mathcal{A}} \neq \emptyset$, and $(r \sqcup s)(d,d') \notin \widehat{\mathcal{A}}$,
then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{(r \sqcup s)(d,d')\}$.

Beside this, we also modify the rule **(4)** in order to handle existential concepts with role union (not just role names).

Meanwhile, for $\mathcal{EL}^*$, we obtain a similar algorithm from the algorithm used for $\mathcal{ELU}$ on page 21 by modifying the rule **(4)** in order to take care of existential concepts with transitive closures, and replacing the rule **(3)** with the following two rules.

(3a) If $r^+ \in \mathsf{rol}(D)$, $r(d,d') \in \widehat{\mathcal{A}}$, and $r^+(d,d') \notin \widehat{\mathcal{A}}$,
then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{r^+(d,d')\}$.

(3b) If $r^+ \in \mathsf{rol}(D)$, $\{r(d,d'), r^+(d',d'')\} \subseteq \widehat{\mathcal{A}}$, and $r^+(d,d'') \notin \widehat{\mathcal{A}}$,
then $\widehat{\mathcal{A}} \coloneqq \widehat{\mathcal{A}} \cup \{r^+(d,d'')\}$.

Like in Lemma 3.5, it is not difficult to show that in both DLs $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^*$, the algorithms decide instance checking without TBoxes and run in time polynomial in the size of the input ABox. Hence, we obtain the following proposition.

Proposition 3.18
Data complexity of instance checking for $\mathcal{EL}^{\cup}$ and $\mathcal{EL}^*$ with simple ABoxes and without TBoxes are polynomial. $\diamond$

3.6 Summary of the Chapter

We end this chapter with a brief summary of all results which has been obtained so far. These results are listed in Table 3.1. In the table, k, k_1 and k_2 are fixed nonnegative integers.

The results of this chapter are about data complexity of instance checking for extensions of $\mathcal{EL}$ with respect to TBoxes. As mentioned in the beginning of the chapter, we intended the results to be established with respect to general TBoxes. But apparently, general TBoxes is not always necessary. With respect to general TBoxes, all DLs considered in this chapter have a coNP-hard (most even coNP-complete) instance checking regarding data complexity, except those DLs which will be discussed in Chapter 4. With respect to acyclic TBoxes, the results are also almost completely mapped out. The only missing results are the coNP lower bound for the DL $\mathcal{EL}^{\geq k}$, $k \geq 3$, as well as the coNP upper bound for $\mathcal{EL}^{R\neg}$, $\mathcal{EL}^{\cup}$, and $\mathcal{EL}^*$. Meanwhile, for the case in which TBoxes are disallowed, the table shows that for some of the extensions of $\mathcal{EL}$ considered here, instance checking is polynomial regarding data complexity. Intractability in this case was established for DLs in which we can express a pair of concepts whose union is equivalent to the top-concept, i.e., the DLs $\mathcal{EL}^{(\neg)}$, $\mathcal{EL}^{\neg}$, $\mathcal{EL}^{\forall r.\bot}$, $\mathcal{EL}^{\forall}$, $\mathcal{EL}^{\leq k_1, \geq k_2}$, $\mathcal{EL}^{\leq k}$, and $\mathcal{EL}^{R\neg}$.

Extensions of $\mathcal{EL}$	Data complexity of inst. checking with simple ABoxes		
	without TBoxes	w.r.t. acyclic TBoxes	w.r.t. general TBoxes
$\mathcal{EL}^{(\neg)}$, $\mathcal{EL}^{\neg}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{ELU}$	in P	coNP-complete	coNP-complete
$\mathcal{EL}^{\forall r.\bot}$, $\mathcal{EL}^{\forall}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\leq k_1, \geq k_2}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\leq k}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{kf}$, $k \geq 2$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\geq 2}$	in P	coNP-complete	coNP-complete
$\mathcal{EL}^{\geq k}$, $k \geq 3$	in P	in coNP, hardness still open	coNP-complete
$\mathcal{EL}^{R\neg}$	coNP-hard	coNP-hard	coNP-hard
$\mathcal{EL}^{\cup}$, $\mathcal{EL}^{*}$	in P	coNP-hard	coNP-hard

Table 3.1: Data complexity of instance checking for various extensions of $\mathcal{EL}$

An exception to this is the DL $\mathcal{EL}^{kf}$, $k \geq 2$, for which data complexity of instance checking without TBoxes is coNP-hard, despite we cannot express a pair of concepts whose union is equivalent to the top-concept.

Chapter 4

A Tractable Extension of $\mathcal{EL}$ Regarding Data Complexity: $\mathcal{ELI}^f$

This chapter deals with an extension of $\mathcal{EL}$ for which data complexity of instance checking w.r.t. general TBoxes has a polynomial upper bound. The extension of $\mathcal{EL}$ which we are concerned with is obtained by adding inverse roles and global (1-)functionality to $\mathcal{EL}$. Recall from Chapter 3, Section 3.4 that global functionality allows some roles to be interpreted as a function everywhere.

In [Baader *et al.*, 2005a,b], it has been shown that subsumption w.r.t. general TBoxes for extensions of $\mathcal{EL}$ with either inverse roles ($\mathcal{ELI}$) or global functionality ($\mathcal{EL}^f$) is ExpTime-complete. This means that regarding combined complexity, instance checking w.r.t. general TBoxes for $\mathcal{ELI}$ and $\mathcal{EL}^f$ has an ExpTime lower bound. However, if we consider data complexity, instance checking w.r.t. general TBoxes for those extensions of $\mathcal{EL}$ can be shown to be tractable. In fact, we can show that tractability is retained even if *both* inverse roles and global functionality are added to $\mathcal{EL}$.

In this chapter, we present the tractability result of instance checking for $\mathcal{ELI}^f$ w.r.t. general TBoxes. The DL $\mathcal{ELI}^f$ is the extension of $\mathcal{EL}$ with inverse roles and global functionality. This tractability result will be proved by providing an algorithm that decides instance checking w.r.t. general TBoxes and runs in time polynomial in the size of the input ABox.

4.1 The Description Logic $\mathcal{ELI}^f$

The DL $\mathcal{ELI}^f$ is obtained by extending $\mathcal{EL}$ with inverse roles and global functionality. Syntax and semantics of inverse roles have been given in Table 2.2. Whereas for global functionality, the reader may refer to Definition 3.9 on page 26. For presentation in this chapter, we express global functionality with GCIs of the form $\top \sqsubseteq (\leq 1\ r)$ for every *role* r which is required to be interpreted as a function. Note that functionality is only in one direction, i.e., if a role name r is functional, then its inverse r^- is not necessarily functional. If we require an inverse role r^- to be functional, then we simply use the GCI $\top \sqsubseteq (\leq 1\ r^-)$ in the TBox.

The algorithm described in this chapter takes four pieces of inputs, namely a general TBox, an ABox, a query concept and an individual name. For this algorithm, we assume that the ABox is simple and the general TBox is normalized appropriately. Thus, we first introduce the notion of normalized TBoxes below.

4.2 A Normal Form for $\mathcal{ELI}^f$ TBoxes

In the following, for a TBox $\mathcal{T}$, we use $\mathsf{BC}_\mathcal{T}$ to denote the set that contains precisely the top-concept $\top$ and all concept names appearing in $\mathcal{T}$. Here, BC stands for *base concepts.*

Definition 4.1 (Normalized $\mathcal{ELI}^f$ TBox)
Let $\mathcal{T}$ be a general $\mathcal{ELI}^f$ TBox. Then $\mathcal{T}$ is *normalized* if every GCI in $\mathcal{T}$ has one of the following forms, where $A_1, A_2, B \in \mathsf{BC}_\mathcal{T}$:

$$\begin{array}{cccc} A_1 \sqsubseteq B, & A_1 \sqsubseteq \exists r.B, & A_1 \sqsubseteq \exists r^-.B & \top \sqsubseteq (\leq 1\, r) \\ A_1 \sqcap A_2 \sqsubseteq B, & \exists r.A_1 \sqsubseteq B, & \exists r^-.A_1 \sqsubseteq B & \top \sqsubseteq (\leq 1\, r^-) \end{array} \quad \diamond$$

By introducing new concept names, any $\mathcal{ELI}^f$ general TBox $\mathcal{T}$ can be transformed into a normalized $\mathcal{ELI}^f$ TBox $\mathcal{T}'$ that is a *conservative extension* of $\mathcal{T}$, i.e., every model of $\mathcal{T}'$ is a model of $\mathcal{T}$ and every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately interpreting additional concept names. As the following lemma shows, this transformation can be done in linear time yielding a normalized TBox whose size is linear in the size of the original TBox.

Lemma 4.2
For $\mathcal{ELI}^f$, instance checking w.r.t. a general TBox $\mathcal{T}$ can be reduced in polynomial time to instance checking w.r.t. a normalized TBox $\mathcal{T}'$ whose size is linear in the size of $\mathcal{T}$.

Proof. A general $\mathcal{ELI}^f$ TBox $\mathcal{T}$ can be converted into a normalized one by using normalization rules (4.1a)–(4.1g) given below, where $B \in \mathsf{BC}_\mathcal{T}$, C, D and E are concepts, $\hat{C}, \hat{D} \notin \mathsf{BC}_\mathcal{T}$, and A is a new concept name.

$$C \sqcap \hat{D} \sqsubseteq E \longrightarrow \{\hat{D} \sqsubseteq A, C \sqcap A \sqsubseteq E\} \tag{4.1a}$$
$$\exists r.\hat{C} \sqsubseteq D \longrightarrow \{\hat{C} \sqsubseteq A, \exists r.A \sqsubseteq D\} \tag{4.1b}$$
$$\exists r^- \hat{C} \sqsubseteq D \longrightarrow \{\hat{C} \sqsubseteq A, \exists r^-.A \sqsubseteq D\} \tag{4.1c}$$

$$\hat{C} \sqsubseteq \hat{D} \longrightarrow \{\hat{C} \sqsubseteq A, A \sqsubseteq \hat{D}\} \tag{4.1d}$$
$$B \sqsubseteq \exists r.\hat{C} \longrightarrow \{B \sqsubseteq \exists r.A, A \sqsubseteq \hat{C}\} \tag{4.1e}$$
$$B \sqsubseteq \exists r^-.\hat{C} \longrightarrow \{B \sqsubseteq \exists r^-.A, A \sqsubseteq \hat{C}\} \tag{4.1f}$$
$$B \sqsubseteq C \sqcap D \longrightarrow \{B \sqsubseteq C, B \sqsubseteq D\} \tag{4.1g}$$

Transformation can be done in linear time resulting in a normalized $\mathcal{ELI}^f$ TBox $\mathcal{T}'$ whose size is linear in $|\mathcal{T}|$ if the normalization rules are applied in two phases:

1. exhaustively apply rules (4.1a), (4.1b) and (4.1c);

2. exhaustively apply rules (4.1d), (4.1e), (4.1f) and (4.1g).

Here, "rule application" means replacing the GCI on the left-hand side with the set of GCIs on the right-hand side. The rule (4.1a) is applied modulo commutativity of conjunction. By examining each rule, it is straightforward to verify that a normalized $\mathcal{ELI}^f$ TBox $\mathcal{T}'$ is obtained by at most $|\mathcal{T}|$ rule applications and $|\mathcal{T}'|$ is linear in $|\mathcal{T}|$. Moreover, it is easy to show that every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately interpreting new concept names introduced by the normalization rules. Since it is clear that every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$, $\mathcal{T}'$ is indeed a conservative extension of $\mathcal{T}$. Thus, given an ABox $\mathcal{A}$, a query concept Q, and an individual d, and provided that Q uses no new concept name from $\mathcal{T}'$ which is introduced during rule applications, it holds that $(\mathcal{T}, \mathcal{A}) \models Q(d)$ iff $(\mathcal{T}', \mathcal{A}) \models Q(d)$. $\diamond$

First, note that the assumption where Q is not allowed to use the new concept names from $\mathcal{T}'$ is really necessary for the correctness of the reduction. But this is not really a problem since we can always apply normalization rules carefully without violating this assumption. In addition, the TBox obtained by rule applications above is of linear size only because the normalization rules are applied in two phases. If they are applied together in one phase, we obtain a quadratic blowup in the worse case due to duplication of the concept B in (4.1g).

4.3 An Instance Checking Algorithm for $\mathcal{ELI}^f$

As already mentioned in the beginning of the chapter, we will derive the tractability of instance checking for $\mathcal{ELI}^f$ regarding data complexity by providing an algorithm that decides instance checking for $\mathcal{ELI}^f$ w.r.t. general TBoxes and runs in time polynomial in the size of the input ABox. In the current section, we will describe this algorithm in more detail. Because in the previous section, we have shown that, for $\mathcal{ELI}^f$, instance checking w.r.t. general TBoxes can be reduced to instance checking w.r.t. normalized TBoxes, we assume from now on that the input TBox for the algorithm is already normalized.

The algorithm which will be described here takes four inputs: a normalized $\mathcal{ELI}^f$ TBox $\mathcal{T}$, a simple ABox $\mathcal{A}_{in}$, a query concept Q and an individual name d. We assume that d is one of the individuals occurring in $\mathcal{A}$, because otherwise, $(\mathcal{T}, \mathcal{A}_{in}) \not\models Q(d)$ trivially holds. W.l.o.g., we also assume that the query concept is a *concept name*, since for every concept D, it is easy to see that $(\mathcal{T}, \mathcal{A}) \models D(d)$ iff $(\mathcal{T} \cup \{A_q \equiv D\}, \mathcal{A}) \models A_q(d)$ where A_q is a new concept name. For the data structure, we use an ABox $\mathcal{A}$ which is initialized to $\mathcal{A}_{in}$. The algorithm is realized in the form of a set of *completion rules* which are used to manipulate $\mathcal{A}$ depending on whether certain preconditions are currently satisfied by $\mathcal{A}$ and $\mathcal{T}$. Since the objective of the algorithm is to decide whether $(\mathcal{T}, \mathcal{A}_{in}) \models Q(d)$, i.e., whether $Q(d)$ is a logical consequence of $\mathcal{T}$ and $\mathcal{A}_{in}$, every completion rule manipulate the ABox $\mathcal{A}$ by generating a new logical consequence of $\mathcal{T}$ and the current $\mathcal{A}$, and adds it to $\mathcal{A}$. For $\mathcal{ELI}^f$, this logical consequence can have two forms: a new concept assertion or an identification of two individual names.

Note that one can impose the so-called *unique name assumption* on $\mathcal{A}$ which requires the mapping from individual names to elements of the domain of an interpretation to be injective. In combination with identifications of individual names, this assumption may

CR1 If $\{r(x,y), A(y)\} \subseteq \mathcal{A}$, $\exists r.A \sqsubseteq B \in \mathcal{T}$, and $B(x) \notin \mathcal{A}$
then $\mathcal{A} \coloneqq \mathcal{A} \cup \{B(x)\}$

CR2 If $\{r(y,x), A(y)\} \subseteq \mathcal{A}$, $\exists r^{-}.A \sqsubseteq B \in \mathcal{T}$, and $B(x) \notin \mathcal{A}$
then $\mathcal{A} \coloneqq \mathcal{A} \cup \{B(x)\}$

CR3 Let $C_x \coloneqq \bigsqcap_{A(x) \in \mathcal{A}} A$.
If $C_x \sqsubseteq_{\mathcal{T}} B$ and $B(x) \notin \mathcal{A}$
then $\mathcal{A} \coloneqq \mathcal{A} \cup \{B(x)\}$

CR4 If $\{r(x,y_1), r(x,y_2)\} \subseteq \mathcal{A}$, and $\top \sqsubseteq (\leq 1\ r) \in \mathcal{T}$
then $\mathcal{A} \coloneqq \mathcal{A}[y_2/y_1]$

CR5 If $\{r(x_1,y), r(x_2,y)\} \subseteq \mathcal{A}$, $\top \sqsubseteq (\leq 1\ r^{-}) \in \mathcal{T}$
then $\mathcal{A} \coloneqq \mathcal{A}[x_2/x_1]$

CR6 If $\{r(y,x), A(y)\} \subseteq \mathcal{A}$, $\{\top \sqsubseteq (\leq 1\ r), A \sqsubseteq \exists r.B\} \subseteq \mathcal{T}$, $B(x) \notin \mathcal{A}$
then $\mathcal{A} \coloneqq \mathcal{A} \cup \{B(x)\}$

CR7 If $\{r(x,y), A(y)\} \subseteq \mathcal{A}$, $\{\top \sqsubseteq (\leq 1\ r^{-}), A \sqsubseteq \exists r^{-}.B\} \subseteq \mathcal{T}$, $B(x) \notin \mathcal{A}$
then $\mathcal{A} \coloneqq \mathcal{A} \cup \{B(x)\}$

Figure 4.1: Completion rules for instance checking in $\mathcal{ELI}^f$

result in an inconsistent ABox which immediately implies any assertions. Therefore, in order not to complicate the discussion, we simply drop this assumption.

The *instance checking algorithm for $\mathcal{ELI}^f$* works as follows. It takes a simple ABox $\mathcal{A}_{in}$, a normalized TBox $\mathcal{T}$, a query concept name A_q, and an individual name d as inputs, and initializes the ABox (for data structure) $\mathcal{A}$ to $\mathcal{A}_{in}$. The algorithm then exhaustively applies all completion rules in Figure 4.1 to $\mathcal{A}$ (in arbitrary order) until no more rule applies. We assume here that for every individual name a occurring in $\mathcal{A}_{in}$, $\top(a) \in \mathcal{A}_{in}$ (otherwise, we simply add such assertions to $\mathcal{A}$ before applying any rules). Note that A and B in the completion rules are concept names or $\top$, and x, y, x_1, x_2, y_1 and y_2 are individual names in $\mathcal{A}$. Let $\mathcal{A}_{out}$ be $\mathcal{A}$ after no more rule applies, and A_q the query concept name. The algorithm returns "yes" if $A_q(d) \in \mathcal{A}_{out}$ and "no" if $A_q(d) \notin \mathcal{A}_{out}$.

For rules **CR1** and **CR2**, the formulation should be clear from the Figure 4.1. Each of them basically adds a new concept name assertion to $\mathcal{A}$, whenever an individual belongs to some existential restriction and this existential restriction implies the concept in the new assertion.

For applying **CR3**, we first need to decide whether $C_x \sqsubseteq_{\mathcal{T}} B$. This can be done by employing a suitable decision procedure for subsumption for any decidable DLs that can express $\mathcal{ELI}^f$, for example, $\mathcal{ALCFI}$. Note that subsumption for such DLs may have complexity that is worse than polynomial, e.g., for $\mathcal{ALCFI}$, subsumption w.r.t. general

TBoxes is ExpTime-complete [Calvanese & De Giacomo, 2003; Donini, 2003]. But the complexity measure of deciding the subsumption $C_x \sqsubseteq_{\mathcal{T}} B$ depends only on the size of $\mathcal{T}$, and not on the size of $\mathcal{A}$. As we aim to characterize the complexity of the algorithm in terms of the size of ABox and not the TBox, which means that the size of the TBox is assumed to be constant, this intractability of subsumption can be safely ignored, i.e., as if the subsumption $C_x \sqsubseteq_{\mathcal{T}} B$ can be decided in constant time.

Rules **CR4** and **CR5** identify two fillers of a functional role. This ensures that functionality restrictions are not violated. We define $\mathcal{A}[x/y]$ as the ABox obtained from $\mathcal{A}$ by replacing all occurrences of the individual name x with the individual name y. In general, if there are two individuals x_1 and x_2 that have to be identified, we can arbitrarily choose between replacing all occurrences of x_2 with x_1, and replacing all occurrences of x_1 with x_2. The only exception is when one of the individual name to be identified is the input individual name d. In this situation, we always replace all occurrences of the other individual name with d. This is done in order to ensure that d still occurs in the ABox $\mathcal{A}$ after no more rule applies.

Rules **CR6** and **CR7** work by propagating concept names which are consequences of TBox axioms to the appropriate fillers of the functional roles. These rules are necessary because whenever an individual a is known to be an instance of an existential restriction $\exists r.B$ with r functional and an individual b is known to be an r-filler of a, then obviously, b must be an instance of the concept B.

4.4 Termination and Soundness

We first prove that the algorithm terminates after a finite number of rule applications which is polynomial in the size of $\mathcal{A}_{in}$. Note that in this analysis, the size of $\mathcal{T}$ is assumed to be constant.

Proposition 4.3 (Termination)
For every normalized $\mathcal{ELI}^f$ TBox $\mathcal{T}$ and simple $\mathcal{ELI}^f$ ABox $\mathcal{A}_{in}$, the rules in Figure 4.1 can only be applied a number of times polynomial in the size of $\mathcal{A}_{in}$, and each rule application can be done in time polynomial in the size of $\mathcal{A}_{in}$.

Proof. It is easily verified that the cardinality of $\mathsf{BC}_{\mathcal{T}}$ is linear in $|\mathcal{T}|$. We first consider applications of the rules **CR1**–**CR3**, **CR6** and **CR7**. Each application of these rules by the algorithm adds a concept assertion for an existing individual name whose concept is a new element of $\mathsf{BC}_{\mathcal{T}}$. None of these rules removes assertions from $\mathcal{A}$. Thus, the number of times that these rules can be performed on a particular individual is linear in the size of $\mathcal{T}$. Since the number of individual names occurring in $\mathcal{A}$ is linear in the size of $\mathcal{A}_{in}$, and no rule adds new individuals to $\mathcal{A}$, the total number of applications of these rules is linear in $|\mathcal{A}_{in}||\mathcal{T}|$. This is linear in $|\mathcal{A}_{in}|$ because the size of $\mathcal{T}$ is considered constant.

For each application of **CR4** and **CR5**, an individual name is removed from $\mathcal{A}$. Since no rule adds new individuals to $\mathcal{A}$, the number of applications of these rules is bounded by the number of individual names in $\mathcal{A}_{in}$ which is linear in the size of $\mathcal{A}_{in}$. Thus, the total number of rule applications is linear in the size of $\mathcal{A}_{in}$.

Note that the size of $\mathcal{A}$ during rule applications is linearly bounded by $|\mathcal{A}_{in}||\mathcal{T}|$, i.e., the size of $\mathcal{A}$ is always linear in the size of $\mathcal{A}_{in}$. It is easily verified that each application of **CR1**, **CR2**, **CR4**–**CR7** can be done in polynomial time in the size of $\mathcal{A}$. For **CR3**, the rule application involves computing the subsumption relation $C_x \sqsubseteq_{\mathcal{T}} B$ which employs a subsumption algorithm w.r.t. general TBoxes for another decidable DL that can express $\mathcal{ELI}^f$. But this computation, albeit possibly worse than polynomial, depends only on the size of the TBox $\mathcal{T}$ and not on the size of the ABox $\mathcal{A}$. Since we consider data complexity which depends only on the size of the ABox, we conclude that the application of **CR3** can be done in polynomial time in the size of ABox $\mathcal{A}$. We thus conclude that each rule application can be done in polynomial time in the size of $\mathcal{A}_{in}$. $\diamond$

Next, we deal with soundness of the algorithm. Let $\mathcal{A}_{out}$ be the ABox $\mathcal{A}$ after the algorithm terminates, i.e., $\mathcal{A}_{out}$ is obtained by the application of the rules in Figure 4.1 to the input TBox $\mathcal{T}$ and the input ABox $\mathcal{A}_{in}$. We show that if the algorithm returns "yes", i.e., the query concept assertion is in $\mathcal{A}_{out}$, then this assertion is indeed implied by $\mathcal{T}$ and $\mathcal{A}_{in}$.

Proposition 4.4 (Soundness)
Let $\mathcal{A}_{in}$ be a simple ABox , $\mathcal{T}$ a normalized TBox, A_q a query concept name, and d an individual name. Let $\mathcal{A}_{out}$ be the ABox that is obtained by the application of the rules in Figure 4.1 to $\mathcal{T}$ and $\mathcal{A}_{in}$. Then we have,

$$A_q(d) \in \mathcal{A}_{out} \textit{ implies } (\mathcal{T}, \mathcal{A}_{in}) \models A_q(d).$$

Proof. In the following, we use $\mathcal{A}_j$, $i = 0, 1, \ldots$, to denote the ABox after the j-th rule application. For showing soundness, it suffices to show that after the j-th rule application (for every $j = 0, 1, \ldots$), $\varphi \in \mathcal{A}_j$ implies $(\mathcal{T}, \mathcal{A}_{in}) \models \alpha$. This will be proved by induction on i in the following. The induction base ($j = 0$) is trivial, since $\mathcal{A}_0$ is initialized to $\mathcal{A}_{in}$. Assume for induction that the claim holds after the j-th rule application (for some $j \geq 0$). To prove the induction step, i.e., that the claim holds after $j + 1$-st rule application, it suffices to show that everything that is added to $\mathcal{A}_j$ on the $j + 1$-st rule application is implied by $\mathcal{T}$ and $\mathcal{A}_{in}$. The $j + 1$-st rule application is always one of **CR1**–**CR7**, therefore, we distinguish seven cases as follows:

CR1: By induction hypothesis, the precondition of **CR1** implies $(\mathcal{T}, \mathcal{A}_{in}) \models r(x, y)$ and $(\mathcal{T}, \mathcal{A}_{in}) \models A(y)$, i.e., for every model $\mathcal{I}$ of T and $\mathcal{A}_{in}$, $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$ and $y^{\mathcal{I}} \in A^{\mathcal{I}}$, for some concept name A, role name r, and individual names x and y occurring in $\mathcal{A}_j$. By the semantics, it follows that $x^{\mathcal{I}} \in (\exists r.A)^{\mathcal{I}}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$ and $\exists r.A \sqsubseteq B \in \mathcal{T}$ for some concept name B, we thus have that $x^{\mathcal{I}} \in B^{\mathcal{I}}$, i.e., $(\mathcal{T}, \mathcal{A}_{in}) \models B(x)$. Hence, the claim holds after applying **CR1** on the $j + 1$-st rule application, since $\mathcal{A}_{j+1} = \mathcal{A}_j \cup \{B(x)\}$ by **CR1**.

CR2: Similar to the **CR1** case.

CR3: By induction hypothesis, the precondition of **CR3** implies $(\mathcal{T}, \mathcal{A}_{in}) \models C_x(x)$ for some individual name x where $C_x \coloneqq \bigsqcap_{A(x) \in \mathcal{A}} A$. Moreover, it follows from the precondition of **CR3** that $\mathcal{T} \models C_x \sqsubseteq B$ for some concept name B. Obviously, this implies that for every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}_{in}$, $x^{\mathcal{I}} \in B^{\mathcal{I}}$, i.e., $(\mathcal{T}, \mathcal{A}_{in}) \models B(x)$. Hence, the claim holds after applying **CR3** on the $j+1$-st rule application, as $\mathcal{A}_{j+1} = \mathcal{A}_j \cup \{B(x)\}$ by **CR3**.

CR4: By induction hypothesis, the precondition of **CR4** implies $(\mathcal{T}, \mathcal{A}_{in}) \models r(x, y_1)$ and $(\mathcal{T}, \mathcal{A}_{in}) \models r(x, y_2)$, i.e., for every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}_{in}$, $(x^{\mathcal{I}}, y_1^{\mathcal{I}}) \in r^{\mathcal{I}}$ and $(x^{\mathcal{I}}, y_2^{\mathcal{I}}) \in r^{\mathcal{I}}$ for some role name r and individual names x, y_1 and y_2 occurring in $\mathcal{A}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$ and $\top \sqsubseteq (\leq 1\ r) \in \mathcal{T}$, $x^{\mathcal{I}}$ must only have one r-filler. Thus, it must be the case that $y_1^{\mathcal{I}} = y_2^{\mathcal{I}}$. Hence, the claim holds after applying **CR4** on the $i+1$-st rule application, since by **CR4**, $\mathcal{A}_{j+1} = \mathcal{A}_j[y_1/y_2]$ or $\mathcal{A}_{j+1} = \mathcal{A}_j[y_2/y_1]$.

CR5: Similar to the **CR4** case.

CR6: By induction hypothesis, the precondition of **CR6** implies that $(\mathcal{T}, \mathcal{A}_{in}) \models r(y, x)$ and $(\mathcal{T}, \mathcal{A}_{in}) \models A(y)$, i.e., for every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}_{in}$, $(y^{\mathcal{I}}, x^{\mathcal{I}}) \in r^{\mathcal{I}}$ and $y^{\mathcal{I}} \in A^{\mathcal{I}}$, for some concept name A, role name r, and individual names x and y occurring in $\mathcal{A}$. Since $\mathcal{I}$ is a model of $\mathcal{T}$ and $\top \sqsubseteq (\leq 1\ r) \in \mathcal{T}$, we have that $x^{\mathcal{I}}$ is the only r-filler of y. Thus, because $A \sqsubseteq \exists r.B \in \mathcal{T}$ for some concept name B, it must be the case that $x^{\mathcal{I}} \in B^{\mathcal{I}}$, i.e., $(\mathcal{T}, \mathcal{A}_{in}) \models B(x)$. Hence, the claim holds after applying **CR6** on the $j+1$-st rule application, since $\mathcal{A}_{j+1} = \mathcal{A}_j \cup \{B(x)\}$ by **CR6**.

CR7: Similar to the **CR6** case.

This finishes the proof of the claim from which the soundness immediately follows (as the algorithm terminates). $\diamond$

4.5 Completeness

After proving soundness in the previous proposition, we now show that the algorithm is complete, i.e., if $(\mathcal{T}, \mathcal{A}_{in}) \models A_q(d)$ then the algorithm returns "yes" (i.e., $A_q(d) \in \mathcal{A}_{out}$ where $\mathcal{A}_{out}$ is the ABox after termination). We show this by proving the contrapositive, namely, if the algorithm returns "no" (i.e., $A_q(d) \notin \mathcal{A}_{out}$), then $(\mathcal{T}, \mathcal{A}_{in}) \not\models A_q(d)$. Hence, from $\mathcal{A}_{out}$ we need to construct a model of $(\mathcal{T}, \mathcal{A}_{in})$ that does not satisfy $A_q(d)$.

Since $\mathcal{A}_{in} \subseteq \mathcal{A}_{out}$, a model of $\mathcal{A}_{out}$ is clearly also a model of $\mathcal{A}_{in}$. But in constructing a model of $\mathcal{T}$, we cannot just consider individual names occurring in $\mathcal{A}_{out}$, because there may be additional individuals which do not explicitly occur in $\mathcal{A}_{out}$, but are nevertheless necessary for a model of $\mathcal{T}$. The idea is to consider every individual name in $\mathcal{A}_{out}$ separately. For each individual name a in $\mathcal{A}_{out}$, we construct a model of $\mathcal{T}$ starting just from the concept assertions for a. This ensures that the additional individuals which do not explicit occur in $\mathcal{A}_{out}$, but are nevertheless necessary for a model of $\mathcal{T}$, are all properly considered. The desired model is then basically obtained by combining all these separately constructed models.

Definition 4.5 (Closed under $\mathcal{T}$-consequence)
Let $\mathcal{T}$ be a normalized TBox and S a set of *concept names*. Then S is *closed under $\mathcal{T}$-consequence* iff for every concept name A,

$$\bigsqcap_{B \in S} B \models_{\mathcal{T}} A \text{ implies } A \in S. \qquad \diamond$$

Note that a set of concept names S is interpreted as a conjunction of its elements. With a slight abuse of notation, we define for every interpretation $\mathcal{I}$, $S^{\mathcal{I}} \coloneqq (\bigsqcap_{A \in S} A)^{\mathcal{I}}$.

For an individual name a occurring in the ABox $\mathcal{A}_{out}$, let $S(a)$ be the set of all concept names A such that $A(a) \in \mathcal{A}_{out}$. Because **CR3** is not applicable anymore after termination, it is clear that $S(a)$ is closed under $\mathcal{T}$-consequence. Our claim is that for each set S that is closed under $\mathcal{T}$-consequence, there is a model $\mathcal{I}$ of S and $\mathcal{T}$, and an individual $d^{\mathcal{I}}$ in the domain of $\mathcal{I}$ such that the concept names which are true at $d^{\mathcal{I}}$ are *precisely* the concept names in S.

Lemma 4.6
Let $\mathcal{T}$ be a normalized $\mathcal{ELI}^f$ TBox and S a set of concept names that is closed under $\mathcal{T}$-consequence. Then there is a model $\mathcal{I}$ of S and $\mathcal{T}$, and an individual $d \in \Delta^{\mathcal{I}}$ such that for all concept name A,

$$A \in S \textit{ iff } d \in A^{\mathcal{I}} \qquad \diamond$$

We now prove this lemma. Let S be a set of concept names and $\mathcal{T}$ a normalized $\mathcal{ELI}^f$ TBox such that S is closed under $\mathcal{T}$-consequence. We construct a (possibly infinite) tree structure from S and $\mathcal{T}$ whose nodes are labeled with sets of subconcepts of S and $\mathcal{T}$. The construction is done in such a way that the tree describes a model of S and $\mathcal{T}$, and additionally, there is a homomorphism, i.e., a structure-preserving mapping, from the tree to every model of S and $\mathcal{T}$. Of course, the notion of homomorphism mentioned before is still unclear, as its precise definition will only be provided later. But the important point is that the existence of such a homomorphism is needed to ensure that at the root of the tree, all occurring concept names belong to S.

Before we proceed, let us first make some notions mentioned in the previous paragraph more precise with the following definitions.

Definition 4.7 (Subconcepts of concepts, sets of concepts and TBoxes)
For an $\mathcal{ELI}^f$ concept C, the set $\mathsf{sub}(C)$ of *subconcepts* of C is inductively defined by:

- $\mathsf{sub}(A) \coloneqq \{A\}$, for $A \in \{\top\} \cup \mathsf{N_C}$;
- $\mathsf{sub}(C \sqcap D) \coloneqq \{C \sqcap D\} \cup \mathsf{sub}(C) \cup \mathsf{sub}(D)$;
- $\mathsf{sub}(\exists\sigma.C) \coloneqq \{\exists\sigma.C\} \cup \mathsf{sub}(C)$ where σ is a role name or an inverse role;
- $\mathsf{sub}((\leq 1\ \sigma)) \coloneqq \{(\leq 1\ \sigma)\}$ where σ is a role name or an inverse role.

If $\hat{S}$ is a set of $\mathcal{ELI}^f$ concepts, then $\mathsf{sub}(\hat{S}) \coloneqq \bigcup_{C \in \hat{S}} \mathsf{sub}(C)$.
For a GCI $D \sqsubseteq D'$, we define $\mathsf{sub}(D \sqsubseteq D') \coloneqq \mathsf{sub}(D) \cup \mathsf{sub}(D')$.
For a (general) TBox $\hat{\mathcal{T}}$, we define the set $\mathsf{sub}_{\hat{\mathcal{T}}}(C)$ (resp. $\mathsf{sub}_{\hat{\mathcal{T}}}(\hat{S})$) of *subconcepts* of a concept C (resp. a set of concepts $\hat{S}$) *w.r.t.* $\hat{\mathcal{T}}$ by:

- $\mathsf{sub}_{\hat{\mathcal{T}}}(C) \coloneqq \mathsf{sub}(C) \cup \{E \mid E \in \mathsf{sub}(D \sqsubseteq D'), D \sqsubseteq D' \in \hat{\mathcal{T}}\}$;
- $\mathsf{sub}_{\hat{\mathcal{T}}}(\hat{S}) \coloneqq \mathsf{sub}(\hat{S}) \cup \{E \mid E \in \mathsf{sub}(D \sqsubseteq D'), D \sqsubseteq D' \in \hat{\mathcal{T}}\}$. $\diamond$

We assume that all existential concepts $\exists\sigma.D$ in $\mathsf{sub}_{\mathcal{T}}(S)$ are linearly ordered where σ is either a role name or an inverse role. Additionally, we assume that there are k existential concepts in $\mathsf{sub}_{\mathcal{T}}(S)$ and that $\phi(i)$ denotes the i-th existential concept in $\mathsf{sub}_{\mathcal{T}}(S)$.

Definition 4.8 (Tree for sets of concepts and TBoxes)
A *tree for S and $\mathcal{T}$* is a $(2^{\mathsf{sub}_{\mathcal{T}}(S)}, R_{S,\mathcal{T}})$-labeled tree $T_{S,\mathcal{T}} = (\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}, \Lambda_{S,\mathcal{T}})$ where $2^{\mathsf{sub}_{\mathcal{T}}(S)}$ is the collection of subsets of $\mathsf{sub}_{\mathcal{T}}(S)$, $R_{S,\mathcal{T}}$ is the set of all role names and inverse roles appearing in $\mathcal{T}$, $\Delta_{S,\mathcal{T}}$ is the set of nodes satisfying $\Delta_{S,\mathcal{T}} \subseteq \{1, \ldots, k\}^*$ with the empty word ϵ as the root, $\Gamma_{S,\mathcal{T}}$ is the set of edges labeled with elements of $R_{S,\mathcal{T}}$, i.e., it consists of triples of the form (v, σ, v') where v, v' are nodes and σ is an element of $R_{S,\mathcal{T}}$, and $\Lambda_{S,\mathcal{T}}$ is a mapping that maps an element of $\Delta_{S,\mathcal{T}}$ to an element of $2^{\mathsf{sub}_{\mathcal{T}}(S)}$. In this tree, ωi denotes a child of a node ω where i corresponds to the i-th existential concept $\phi(i) = \exists\sigma.A \in \mathsf{sub}_{\mathcal{T}}(S)$, and the edge from ω to ωi is labeled with σ. $\diamond$

The aforementioned notion of homomorphism from a tree for S and $\mathcal{T}$ to a model of S and $\mathcal{T}$ is defined more precisely in the following.

Definition 4.9 (Homomorphism)
Let $T_{S,\mathcal{T}}$ be a tree for S and $\mathcal{T}$, and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ a model of S and $\mathcal{T}$. A *homomorphism* from $T_{S,\mathcal{T}}$ to $\mathcal{J}$ is a mapping τ from the set of nodes $\Delta_{S,\mathcal{T}} \subseteq \{1, \ldots, k\}^*$ to $\Delta^{\mathcal{J}}$ such that for every node $\omega \in \Delta_{S,\mathcal{T}}$:

(h1) $A \in \Lambda_{S,\mathcal{T}}(\omega)$ implies $\tau(\omega) \in A^{\mathcal{I}}$ for every concept name A;

(h2) if $(\omega, r, \omega') \in \Gamma_{S,\mathcal{T}}$ then $(\tau(\omega), \tau(\omega')) \in r^{\mathcal{I}}$.

(h3) if $(\omega, r^-, \omega') \in \Gamma_{S,\mathcal{T}}$ then $(\tau(\omega'), \tau(\omega)) \in r^{\mathcal{I}}$. $\diamond$

Having defined all needed notions, we can now start proving Lemma 4.6. We begin by constructing a tree for S and $\mathcal{T}$ that describes a model of S and $\mathcal{T}$. We then show that for every model of S and $\mathcal{T}$, there is a homomorphism from the tree to it. Afterwards, we show that all concept names occurring at the root of the tree are the ones which precisely belong to S.

We construct the tree by inductively building $\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}$ and $\Lambda_{S,\mathcal{T}}$ starting from the root. For the construction, induction is done on $n \in \mathbb{N}$. We use V_n, E_n and Λ_n to denote $\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}$ and $\Lambda_{S,\mathcal{T}}$ in the n-th step of the construction.

We start with $n = 0$, for which we define:

$$\Delta_0 \coloneqq \{\epsilon\} \qquad\qquad \Gamma_0 \coloneqq \emptyset \qquad\qquad \Lambda_0(\epsilon) \coloneqq S \cup \{\top\}$$

Next, assume that up to some $n \geq 0$, Δ_n, Γ_n are already defined and $\Lambda_n(v)$ is also already defined for all $v \in \Delta_n$. We choose a node $\omega \in \Delta_n$ for which at least one of the following cases are applicable and $|\omega|$ is *minimal*, and proceed with applying any one of the applicable cases. If there is no such node then we simply set $\Delta_{n+1} \coloneqq \Delta_n$, $\Gamma_{n+1} \coloneqq \Gamma_n$ and $\Lambda_{n+1} \coloneqq \Lambda_n$. In the following, σ denotes either a role name or an inverse role, and $\mathsf{Inv}(\sigma)$ denotes the inverse of σ that is defined as:

$$\mathsf{Inv}(\sigma) = \begin{cases} r^- & \text{if } \sigma = r \text{ is a role name} \\ r & \text{if } \sigma = r^- \text{ is an inverse role} \end{cases}$$

(T1) If for some $i \in \{1, \ldots, k\}$, there is an existential concept $\phi(i) = \exists\sigma.A \in \Lambda_n(\omega)$ such that $\top \sqsubseteq (\leq 1\ \sigma) \notin \mathcal{T}$, $(\omega, \sigma, \omega i) \notin \Gamma_n$ and $\omega i \notin \Delta_n$, then $\Delta_{n+1} \coloneqq \Delta_n \cup \{\omega i\}$, $\Gamma_{n+1} \coloneqq \Gamma_n \cup \{(\omega, \sigma, \omega i)\}$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \{A, \top\} & \text{if } \omega' = \omega i, \\ \Lambda_n(\omega') & \text{otherwise} \end{cases}$$

(T2) If for some $i \in \{1, \ldots, k\}$, there is an existential concept $\phi(i) = \exists\sigma.A \in \Lambda_n(\omega)$ such that $\top \sqsubseteq (\leq 1\ \sigma) \in \mathcal{T}$, then we distinguish three subcases.

a) There is a $\hat{\omega} \in \Delta_n$ such that $(\hat{\omega}, \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$ and $A \notin \Lambda_n(\hat{\omega})$.
Then $\Delta_{n+1} \coloneqq \Delta_n$, $\Gamma_{n+1} \coloneqq \Gamma_n$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \Lambda_n(\omega') \cup \{A, \top\} & \text{if } \omega' = \hat{\omega}, \\ \Lambda_n(\omega') & \text{otherwise} \end{cases}$$

b) For every $\hat{\omega} \in \Delta_n$, $(\hat{\omega}, \mathsf{Inv}(\sigma), \omega) \notin \Gamma_n$, but there exists $\omega j \in \Delta_n$ such that $j \neq i$, $(\omega, \sigma, \omega j) \in \Gamma_n$ and $A \notin \Lambda_n(\omega j)$. Then $\Delta_{n+1} \coloneqq \Delta_n$, $\Gamma_{n+1} \coloneqq \Gamma_n$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \Lambda_n(\omega') \cup \{A, \top\} & \text{if } \omega' = \omega j, \\ \Lambda_n(\omega') & \text{otherwise} \end{cases}$$

c) For every $\hat{\omega} \in \Delta_n$, $(\hat{\omega}, \mathsf{Inv}(\sigma), \omega) \notin \Gamma_n$, $(\omega, \sigma, \omega i) \notin \Gamma_n$ and $\omega i \notin \Delta_n$. Then $\Delta_{n+1} \coloneqq \Delta_n \cup \{\omega i\}$, $\Gamma_{n+1} \coloneqq \Gamma_n \cup \{(\omega, \sigma, \omega i)\}$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \{A, \top\} & \text{if } \omega' = \omega i, \\ \Lambda_n(\omega') & \text{otherwise} \end{cases}$$

(T3) If there is a subset $\hat{S} \subseteq \Lambda_n(\omega)$ such that $\hat{S}$ contains *concept names* or the top-concept, $(\bigsqcap_{A \in \hat{S}} A) \sqsubseteq C \in \mathcal{T}$, and $C \notin \Lambda_n(\omega)$, then $\Delta_{n+1} \coloneqq \Delta_n$, $\Gamma_{n+1} \coloneqq \Gamma_n$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \Lambda_n(\omega') \cup \{C, \top\} & \text{if } \omega' = \omega, \\ \Lambda_n(\omega') & \text{otherwise.} \end{cases}$$

(T4) If there is a node $\hat{\omega} \in \Delta_n$ such that either $(\omega, \sigma, \hat{\omega}) \in \Gamma_n$ or $(\hat{\omega}, \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$, and it also holds that $\exists\sigma.A \sqsubseteq B \in \mathcal{T}$, $A \in \Lambda_n(\hat{\omega})$ and $B \notin \Lambda_n(\omega)$, then $\Delta_{n+1} \coloneqq \Delta_n$, $\Gamma_{n+1} \coloneqq \Gamma_n$, and for every $\omega' \in \Delta_{n+1}$,

$$\Lambda_{n+1}(\omega') \coloneqq \begin{cases} \Lambda_n(\omega') \cup \{B, \top\} & \text{if } \omega' = \omega, \\ \Lambda_n(\omega') & \text{otherwise.} \end{cases}$$

The resulting tree $T_{S,\mathcal{T}} = (\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}, \Lambda_{S,\mathcal{T}})$ is obtained by taking unions over all n:

$$\Delta_{S,\mathcal{T}} \coloneqq \bigcup_{n=0}^{\infty} \Delta_n \qquad \Gamma_{S,\mathcal{T}} \coloneqq \bigcup_{n=0}^{\infty} \Gamma_n \qquad \Lambda_{S,\mathcal{T}} \coloneqq \bigcup_{n=0}^{\infty} \Lambda_n$$

By construction, it is trivial to see that $\Delta_{S,\mathcal{T}} \subseteq \{1, \ldots, k\}^*$, $\Gamma_{S,\mathcal{T}} \subseteq \Delta_{S,\mathcal{T}} \times R_{S,\mathcal{T}} \times \Delta_{S,\mathcal{T}}$, and $\Lambda_{S,\mathcal{T}}$ is indeed a mapping from $\Delta_{S,\mathcal{T}}$ to $2^{\mathsf{sub}_{\mathcal{T}}(S)}$. The tree shape is guaranteed because by construction, each node ω (except ϵ) has one predecessor ω' where $\omega = \omega' i$ for some $i \in \{1, \ldots, k\}$.

Notice that this tree is finitely branching but may be infinite due to **(T1)** and **(T2c)** of the inductive construction step which generate new nodes. In addition, each induction step adds one new element from $\mathsf{sub}_{\mathcal{T}}(S)$ to a node label and never removes such elements. Because $\mathsf{sub}_{\mathcal{T}}(S)$ is finite, obviously each node has a finite set as label. Thus, for each node ω, **(T1)**–**(T4)** can only be applied to ω finitely many times. From this observation and the fact that in every construction step, we always choose a node ω for which the length of ω is minimal, it follows that for each node ω, there is a natural number n such that for every $m \geq n$, none of **(T1)**–**(T4)** is applicable in the m-th step of the construction.

We now show that $T_{S,\mathcal{T}} = (\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}, \Lambda_{S,\mathcal{T}})$ describes a model of S and $\mathcal{T}$. We define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with the domain $\Delta^{\mathcal{I}} \coloneqq \Delta_{S,\mathcal{T}}$, and set

$$\begin{aligned} r^{\mathcal{I}} \coloneqq\ & \{(\omega, \omega i) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (\omega, r, \omega i) \in \Gamma_{S,\mathcal{T}}, i \in \{1, \ldots, k\}\} \\ & \cup \{(\omega i, \omega) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (\omega, r^-, \omega i) \in \Gamma_{S,\mathcal{T}}, i \in \{1, \ldots, k\}\} \text{ for each role name } r \\ A^{\mathcal{I}} \coloneqq\ & \{\omega \in \Delta^{\mathcal{I}} \mid A \in \Lambda_{S,\mathcal{T}}(\omega)\} \text{ for each concept name } A \end{aligned}$$

Lemma 4.10
$\mathcal{I}$ is a model of S and $\mathcal{T}$

Proof. Since by construction, the set of concept names S satisfies $S \subseteq \Lambda_{S,\mathcal{T}}(\epsilon)$, we obviously have $\epsilon \in S^{\mathcal{I}}$, i.e., $\mathcal{I}$ is a model of S. It now remains to show that $\mathcal{I}$ is also a model of $\mathcal{T}$. We examine each possible form of GCI in $\mathcal{T}$ which is already normalized.

- $\top \sqsubseteq (\leq 1\ r)$ and $\top \sqsubseteq (\leq 1\ s^-)$. GCIs of these forms are satisfied since by **(T2)**, every $\omega \in \Delta_{S,\mathcal{T}}$ obeys all functionality restrictions.
- $A \sqsubseteq C$ where A is a concept name or $\top$, and C is either a concept name or an existential concept. Suppose $\omega \in A^{\mathcal{I}}$. By definition of $\mathcal{I}$, $A \in \Lambda_{S,\mathcal{T}}(\omega)$. Because after finitely many construction steps, none of **(T1)**–**(T4)** is applicable to ω, **(T3)** has already been

applied to ω. Hence, we have $C \in \Lambda_{S,\mathcal{T}}(\omega)$. If C is a concept name, then the definition of $\mathcal{I}$ implies that $\omega \in C^{\mathcal{I}}$. If C is an existential concept, then $\omega \in C^{\mathcal{I}}$ is an immediate consequence of the definition of $\mathcal{I}$ and the fact that either **(T1)** or **(T2)** has already been applied to ω.

- $A_1 \sqcap A_2 \sqsubseteq B$ where A_1, A_2 and B are concept names or $\top$. Assume $\omega \in (A_1 \sqcap A_2)^{\mathcal{I}} = A_1^{\mathcal{I}} \cap A_2^{\mathcal{I}}$. By definition of $\mathcal{I}$, $\{A_1, A_2\} \subseteq \Lambda_{S,\mathcal{T}}(\omega)$. Since after finitely many construction steps, none of **(T1)**–**(T4)** is applicable, **(T3)** must already have been applied to ω. Consequently, $B \in \Lambda_{S,\mathcal{T}}(\omega)$. By definition of $\mathcal{I}$, we conclude $\omega \in B^{\mathcal{I}}$.

- $\exists r.A \sqsubseteq B$ where A and B are concept names or $\top$. Suppose $\omega \in (\exists r.A)^{\mathcal{I}}$. Then there is an $\omega' \in \Delta^{\mathcal{I}}$ such that $(\omega, \omega') \in r^{\mathcal{I}}$ and $\omega' \in A^{\mathcal{I}}$ which implies $A \in \Lambda_{S,\mathcal{T}}(\omega')$ by definition of $\mathcal{I}$. By definition of $r^{\mathcal{I}}$, either $\omega' = \omega i$ or $\omega = \omega' i$ for some $i \in \{1, \ldots, k\}$, which implies either $(\omega, r, \omega') \in \Gamma_{S,\mathcal{T}}$ or $(\omega', r^-, \omega) \in \Gamma_{S,\mathcal{T}}$. For both cases, because **(T1)**–**(T4)** are no longer applicable after finitely many construction steps, **(T4)** has been applied to ω. This yields $B \in \Lambda_{S,\mathcal{T}}(\omega)$. Consequently, by definition of $\mathcal{I}$, $\omega \in B^{\mathcal{I}}$.

- $\exists r^-.A \sqsubseteq B$ where A and B are concept names or $\top$. Similar to the previous case.

Since $\mathcal{I}$ satisfies all possible form of GCIs in a normalized $\mathcal{T}$, we conclude that $\mathcal{I}$ is indeed a model of $\mathcal{T}$. $\diamond$

Next, we show in the following lemma that for every model of S and $\mathcal{T}$, there is a homomorphism from the tree $T_{S,\mathcal{T}}$ to it.

Lemma 4.11
For every model $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ of S and $\mathcal{T}$, there exists a homomorphism from $T_{S,\mathcal{T}} = (\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}, \Lambda_{S,\mathcal{T}})$ to $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$.

Proof. Let $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ be any arbitrary model of S and $\mathcal{T}$. We construct an appropriate mapping $\tau \colon \Delta_{S,\mathcal{T}} \mapsto \Delta^{\mathcal{J}}$. The construction is done inductively according to the inductive construction of $T_{S,\mathcal{T}}$. We will use τ_n to denote the mapping in the n-th step of the construction where $n = 0, 1, \ldots$. We also use the notation $(\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T1)},\, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1}), \ldots, (\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T4)},\, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$ to indicate that the $n+1$-st construction step is due to applying **(T1)**–**(T4)** to a node $\omega \in \Delta_n$. During the construction, we maintain the following property for every $n = 0, 1, \ldots$:

(P1) for every concept name A, $A \in \Lambda_n(\omega)$ implies $\tau_n(\omega) \in A^{\mathcal{J}}$ for all $\omega \in \Delta_n$;

(P2) for every role name r and nodes $\omega, \omega' \in \Delta_n$, $(\omega, r, \omega') \in \Gamma_n$ implies $(\tau_n(\omega), \tau_n(\omega')) \in r^{\mathcal{J}}$;

(P3) for every role name r and nodes $\omega, \omega' \in \Delta_n$, $(\omega, r^-, \omega') \in \Gamma_n$ implies $(\tau_n(\omega'), \tau_n(\omega)) \in r^{\mathcal{J}}$.

We start from $n = 0$, beginning with the root ϵ. By the construction of the tree, $\Delta_0 = \{\epsilon\}$, $\Gamma_0 = \emptyset$ and $\Lambda_0 = \{(\epsilon, S \cup \{\top\})\}$. Since $\mathcal{J}$ is a model of S, there is an individual $x_0 \in \Delta^{\mathcal{J}}$ such that $x_0 \in S^{\mathcal{J}}$. We thus simply choose such an x_0 and set $\tau_0(\epsilon) \coloneqq x_0$, i.e., $\tau_0 \coloneqq \{(\epsilon, x_0)\}$. Note that **(P1)**–**(P3)** hold for $n = 0$.

Assume now for induction that **(P1)**–**(P3)** hold for some $n \geq 0$. We show that **(P1)**–**(P3)** also hold for $n + 1$ while at the same time constructing τ_{n+1} from τ_n. First of all, if $(\Delta_n, \Gamma_n, \Lambda_n) = (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$ (no change in the tree), then we simply set $\tau_{n+1} \coloneqq \tau_n$, whereby it is obvious that **(P1)**–**(P3)** are still satisfied. Otherwise, we make case distinction based on cases of the tree construction. Here, σ denotes a role name or an inverse role.

- $(\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T1)},\, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$. Due to **(T1)**, for some $i \in \{1, \ldots, k\}$, there is an existential concept $\phi(i) = \exists\sigma.A \in \Lambda_n(\omega)$, such that $\top \sqsubseteq (\leq 1\ \sigma) \notin \mathcal{T}$, $(\omega, \sigma, \omega i) \notin \Gamma_n$ and $\omega i \notin \Delta_n$, and we have $\Delta_{n+1} = \Delta_n \cup \{\omega i\}$, $\Gamma_{n+1} = \Gamma_n \cup \{(\omega, \sigma, \omega i)\}$ and $\Lambda_{n+1} = \Lambda_n \cup \{(\omega i, \{A, \top\})\}$. Since **(P1)** is satisfied for n, we have $\tau_n(\omega) \in (\exists\sigma.A)^{\mathcal{J}}$. By the semantics, there is an individual $x \in \Delta^{\mathcal{J}}$ such that $(\tau_n(\omega), x) \in \sigma^{\mathcal{J}}$ and $x \in A^{\mathcal{J}}$. Hence, we choose such an x and set $\tau_{n+1} \coloneqq \tau_n \cup \{(\omega i, x)\}$, i.e., $\tau_{n+1}(\omega i) \coloneqq x$. By this construction step, **(P1)**, **(P2)** and **(P3)** clearly still hold for $n + 1$.

- $(\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T2)},\, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$. By **(T2)**, for some $i \in \{1, \ldots, k\}$, there is an existential concept $\phi(i) = \exists\sigma.A \in \Lambda_n(\omega)$ such that $\top \sqsubseteq (\leq 1\ \sigma) \in \mathcal{T}$ and precisely one of the following holds:

  ⋆ there is an $\omega' \in \Delta_n$ such that $(\omega', \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$ and $A \notin \Lambda_n(\omega')$ (by **(T2a)**);

  ⋆ there is no $\omega' \in \Delta_n$ such that $(\omega', \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$ but there is an $\omega j \in \Delta_n$ such that $j \neq i$ and $(\omega, \sigma, \omega j) \in \Gamma_n$ and $A \notin \Lambda_n(\omega j)$ (by **(T2b)**);

  ⋆ neither $(\omega', \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$ nor $(\omega, \sigma, \omega') \in \Gamma_n$ for every $\omega' \in \Delta_n$ (by **(T2c)**).

  In all three cases, we have that $\tau_n(\omega) \in (\exists\sigma.A)^{\mathcal{J}}$ and $\tau_n(\omega) \in (\leq 1\ \sigma)^{\mathcal{J}}$ since **(P1)** is satisfied for n and $\mathcal{J}$ is a model of $\mathcal{T}$. Thus, there is a unique individual $x \in \Delta^{\mathcal{J}}$ such that $(\tau_n(\omega), x) \in \sigma^{\mathcal{J}}$ and $x \in A^{\mathcal{J}}$.

  For the first case, no new node and edge are added. The only change is $\Lambda_{n+1}(\omega') = \Lambda_n(\omega') \cup \{A\}$. Since **(P3)** is satisfied for n, we already have $(\tau_n(\omega), \tau_n(\omega')) \in \sigma^{\mathcal{J}}$. Moreover, $\tau_n(\omega')$ is the unique σ-filler of $\tau_n(\omega)$ because $\tau_n(\omega) \in (\leq 1\ \sigma)^{\mathcal{J}}$ as $\mathcal{J}$ is a model of $\mathcal{T}$. Thus, it must be the case that $\tau_n(\omega') = x$ is satisfied before the current induction step. As no change is needed to τ_n, we simply set $\tau_{n+1} \coloneqq \tau_n$. Note that **(P1)**, **(P2)** and **(P3)** clearly hold for $n+1$ since they hold for n and adding A to the label of ω' is the only change.

  The second case is very similar to the first one where the only change is $\Lambda_{n+1}(\omega j) = \Lambda_n(\omega j) \cup \{A\}$ with $j \neq i$. As **(P2)** is satisfied for n, we have $(\tau_n(\omega), \tau_n(\omega j)) \in \sigma^{\mathcal{J}}$. Moreover, $\tau_n(\omega j)$ is the unique σ-filler for $\tau_n(\omega)$. Thus, $\tau_n(\omega j) = x$ already holds before the current induction step. Thus, like in the first case, we simply set $\tau_{n+1} \coloneqq \tau_n$. For $n + 1$, **(P1)**, **(P2)**, and **(P3)** hold similarly like the first case above.

For the third case, we have added a new node ωi, a new edge $(\omega, \sigma, \omega i)$ and set the label of ωi with $\Lambda_{n+1}(\omega i) = \{A, \top\}$. For the construction of τ_{n+1}, we simply set $\tau_{n+1} := \tau_n \cup \{(\omega i, x)\}$, i.e., $\tau_{n+1}(\omega i) := x$. Like application of **(T1)**, we have that **(P1)**, **(P2)** and **(P3)** hold for $n+1$.

- $(\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T3)}, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$. Applying **(T3)** means there is a subset $\hat{S}$ of $\Lambda_n(\omega)$ containing concept names or $\top$ such that $(\bigsqcap_{A \in \hat{S}} A) \sqsubseteq D \in \mathcal{T}$ where $D \notin \Lambda_n(\omega)$, and either $D \in \mathsf{BC}_{\mathcal{T}}$ or D is an existential concept. Moreover, we have that $\Delta_{n+1} = \Delta_n$, $\Gamma_{n+1} = \Gamma_n$ and $\Lambda_{n+1} = (\Lambda_n \setminus \{(\omega, \Lambda_n(\omega))\}) \cup \{(\omega, \Lambda_n(\omega) \cup \{D\})\}$. In this case, we just set $\tau_{n+1} := \tau_n$. By construction of τ_{n+1}, induction, and the fact that $\mathcal{J}$ is a model of $\mathcal{T}$, **(P1)**, **(P2)** and **(P3)** obviously hold for $n+1$.

- $(\Delta_n, \Gamma_n, \Lambda_n) \xrightarrow{\textbf{(T4)}, \omega} (\Delta_{n+1}, \Gamma_{n+1}, \Lambda_{n+1})$. There is a node $\hat{\omega} \in \Delta_n$ such that either $(\omega, \sigma, \hat{\omega}) \in \Gamma_n$ or $(\hat{\omega}, \mathsf{Inv}(\sigma), \omega) \in \Gamma_n$, and it also holds that $\exists \sigma.A \sqsubseteq B \in \mathcal{T}$, $A \in \Lambda_n(\hat{\omega})$ and $B \notin \Lambda_n(\omega)$ for $A, B \in \mathsf{BC}_{\mathcal{T}}$. Furthermore, we have $\Delta_{n+1} = \Delta_n$, $\Gamma_{n+1} = \Gamma_n$ and $\Lambda_{n+1} = (\Lambda_n \setminus \{(\omega, \Lambda_n(\omega))\}) \cup \{(\omega, \Lambda_n(\omega) \cup \{B\})\}$. For this case, we set $\tau_{n+1} := \tau_n$. By construction of τ_{n+1}, induction, and the fact that $\mathcal{J}$ is a model of $\mathcal{T}$, **(P1)**, **(P2)** and **(P3)** obviously hold for $n+1$.

Thus, we conclude that **(P1)**–**(P3)** hold for every $n = 0, 1, \ldots$. The mapping τ is then defined as:

$$\tau := \bigcup_{n=0}^{\infty} \tau_n$$

By construction, τ is indeed a well-defined mapping from $\Delta_{S,\mathcal{T}}$ to $\Delta^{\mathcal{J}}$. In addition, as **(P1)**–**(P3)** hold for every $n = 0, 1, \ldots$, **(h1)**–**(h3)** of homomorphism (Definition 4.9) are satisfied by τ, i.e., τ is a homomorphism from $T_{S,\mathcal{T}}$ to a model $\mathcal{J}$ of S and $\mathcal{T}$. Since the proof is independent of the choice of $\mathcal{J}$, we conclude that for every model of S and $\mathcal{T}$, there is always such a homomorphism τ from the tree $T_{S,\mathcal{T}}$ to it. $\diamond$

Next, we use the previous lemma to show that at the root ϵ of the tree $T_{S,\mathcal{T}}$, if a concept name is in its label then the concept name belongs to S.

Lemma 4.12
For every concept name A, $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ implies $A \in S$.

Proof. We show that for every concept name A, $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ implies $S \models_{\mathcal{T}} A$. Suppose otherwise, i.e., that $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ but $S \not\models_{\mathcal{T}} A$. Then there is a model $\mathcal{J}$ of S and $\mathcal{T}$ and an individual $y \in \Delta^{\mathcal{J}}$ such that $y \in S^{\mathcal{J}}$ but $y \notin A^{\mathcal{J}}$. By Lemma 4.11, there is a homomorphism τ from $T_{S,\mathcal{T}}$ to $\mathcal{J}$ such that $\tau(\epsilon) = y$. But due to **(h1)** of homomorphism, $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ implies $y = \tau(\epsilon) \in A^{\mathcal{J}}$ which is a contradiction. Hence, for every concept name A, $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ implies $S \models_{\mathcal{T}} A$. Finally, the lemma is established since S is closed under $\mathcal{T}$-consequence. $\diamond$

Lemma 4.6 is then established as follows. By definition of $T_{S,\mathcal{T}} = (\Delta_{S,\mathcal{T}}, \Gamma_{S,\mathcal{T}}, \Lambda_{S,\mathcal{T}})$ and using Lemma 4.10, we obtain a model $\mathcal{I}$ of S and $\mathcal{T}$ whose shape is a tree with ϵ

as its root and whose domain is all nodes of $T_{S,\mathcal{T}}$, i.e., the set $\Delta_{S,\mathcal{T}}$. Lemma 4.10 also shows that for the root ϵ and every concept name A, $A \in S \subseteq \Lambda_{S,\mathcal{T}}(\epsilon)$ implies $\epsilon \in A^{\mathcal{I}}$. In addition, it follows from Lemma 4.10 and Lemma 4.12 that for every concept name A, $\epsilon \in A^{\mathcal{I}}$ implies $A \in \Lambda_{S,\mathcal{T}}(\epsilon)$ which then implies $A \in S$. Hence, we conclude for every concept name A, that $A \in S$ iff $\epsilon \in A^{\mathcal{I}}$. The existence of the individual d is thus guaranteed by simply setting $d \coloneqq \epsilon$.

The completeness is proved by showing that $(\mathcal{T}, \mathcal{A}_{in}) \models A_q(d)$ implies that the algorithm returns "yes", i.e., $A_q(d) \in \mathcal{A}_{out}$. We show this result by proving the contrapositive, i.e., $A_q(d) \notin \mathcal{A}_{out}$ implies $(\mathcal{T}, \mathcal{A}_{in}) \not\models A_q(d)$. The proof constructs a model, say $\hat{\mathcal{I}}$, of $(\mathcal{T}, \mathcal{A}_{in})$ that does not satisfy $A_q(d)$, provided that $A_q(d) \notin \mathcal{A}_{out}$.

The idea is that a part of $\hat{\mathcal{I}}$ is described by the ABox $\mathcal{A}_{out}$, because a model of $\mathcal{A}_{out}$ is clearly also a model of $\mathcal{A}_{in}$. The other parts of $\hat{\mathcal{I}}$ are obtained by considering additional individuals which do not explicitly occur in $\mathcal{A}_{out}$, but are nevertheless necessary for a model of $\mathcal{T}$ because of the computation of subsumption, i.e., $\mathcal{T}$-consequences, in the rule **CR3**. Such parts of $\hat{\mathcal{I}}$ are constructed by applying Lemma 4.6 to every individual name occurring in $\mathcal{A}_{out}$. $\hat{\mathcal{I}}$ is then obtained by combining all those mentioned parts together. The combination must be done carefully, in order not to violate functional restrictions applied to some roles (role names or inverse roles). In particular, we have to take into account the following situation in which there is a role assertion $f(a, b) \in \mathcal{A}_{out}$ for which f is functional, but on the other hand, an f-filler of a has also been generated due to Lemma 4.6. In such a situation, in order to obey the functionality restrictions, we need to 'cut' a part of the model $\mathcal{T}$ which has been constructed by Lemma 4.6 where this part is connected from a by the functional role f.

We are now ready for the main result of this subsection, where the above idea will be made precise.

Proposition 4.13 (Completeness)
Let $\mathcal{A}_{in}$ be a simple ABox , $\mathcal{T}$ a normalized TBox, A_q a query concept name, and d an individual name. Let $\mathcal{A}_{out}$ be the ABox that is obtained by the application of the rules in Figure 4.1 to $\mathcal{T}$ and $\mathcal{A}_{in}$. Then we have,

$$A_q(d) \notin \mathcal{A}_{out} \text{ implies } (\mathcal{T}, \mathcal{A}_{in}) \not\models A_q(d).$$

Proof. Suppose $A_q(d) \notin \mathcal{A}_{out}$. We construct a model of $(\mathcal{T}, \mathcal{A}_{in})$ that does not satisfy $A_q(d)$.

Assume w.l.o.g. that there are m individual names $b_1, \ldots, b_m$ occurring in $\mathcal{A}_{in}$, there are n individual names $a_1, \ldots, a_n$ occurring in $\mathcal{A}_{out}$ and $d = a_1$. Note that $d \in \{a_1, \ldots, a_n\} \subseteq \{b_1, \ldots, b_m\}$ and $n \leq m$ due to applications of rules **CR4** and **CR5** which identify some individual names. For each a_i, $i = 1, \ldots, n$, we define a set $P(a_i) \subseteq \{b_1, \ldots, b_m\}$ that satisfies

(1) $a_i \in P(a_i)$; and

(2) $b_j \in P(a_i)$ iff there is a $b \in P(a_i)$ such that b_j was identified with b by **CR4** or **CR5**.

It is easy to see that $P(a_i)$, $i = 1, \ldots, n$, form partitions of the set of individuals $\{b_1, \ldots, b_m\}$ in $\mathcal{A}_{in}$. Since **CR4** and **CR5** are sound, if an individual name x is identified with an individual name y, then $x^{\mathcal{J}} = y^{\mathcal{J}}$ for every model $\mathcal{J}$ of $\mathcal{T}$ and $\mathcal{A}_{in}$. Thus, all elements of $P(a_i)$ are interpreted as the same individual in the domain of any model of $\mathcal{T}$ and $\mathcal{A}_{in}$.

For every individual name a_i, $i = 1, \ldots, n$, we define a set S_i as follows:

$$S_i \coloneqq \{A \mid A(a_i) \in \mathcal{A}_{out}\}$$

Since rule **CR3** is not applicable (to $\mathcal{A}_{out}$) anymore after termination, it is clear that S_i is closed under $\mathcal{T}$-consequence. Thus, by Lemma 4.6, for every a_i, $i = 1, \ldots, n$, there is a tree model $\mathcal{I}_i$ of S_i and $\mathcal{T}$ with $a_i^{\mathcal{I}_i}$ as its root (by simply mapping a_i to the root of the tree model $\mathcal{I}_i$) such that for every concept name A, $A \in S_i$ iff $a_i^{\mathcal{I}_i} \in A^{\mathcal{I}_i}$.

Due to Lemma 4.10, each model $\mathcal{I}_i$ of S_i and $\mathcal{T}$ is actually defined such that $\Delta^{\mathcal{I}_i} \subseteq \{1, \ldots, k\}^*$ where k is the number of existential concepts in $\mathsf{sub}_{\mathcal{T}}(S)$. To distinguish such models due to different a_i's, we attach a subscript a_i for every element of $\Delta^{\mathcal{I}_i}$. Thus, ϵ_{a_i} is the root of tree model $\mathcal{I}_i$ where $a_i^{\mathcal{I}_i} = \epsilon_{a_i}$ and by using subscripts here, we can assume that $\Delta^{\mathcal{I}_{j_1}} \cap \Delta^{\mathcal{I}_{j_2}} = \emptyset$ for every $1 \le j_1 < j_2 \le n$.

With every individual name a_i, $i = 1, \ldots, n$, occurring in $\mathcal{A}_{out}$, we associate the set

$$\begin{aligned} F_i \coloneqq \{r \mid \exists b\colon r(a_i, b) \in \mathcal{A}_{out}, \top \sqsubseteq (\le 1\ r) \in \mathcal{T}\} \cup \\ \{s^- \mid \exists b\colon s(b, a_i) \in \mathcal{A}_{out}, \top \sqsubseteq (\le 1\ s^-) \in \mathcal{T}\} \end{aligned}$$

Observe that F_i consists of roles (role names and inverse roles) that connects a_i to another individual in $\mathcal{A}_{out}$ and is required to be functional.

For each $\mathcal{I}_i$, $i = 1, \ldots, n$, we define an interpretation $\mathcal{J}_i = (\Delta^{\mathcal{J}_i}, \cdot^{\mathcal{J}_i})$ from the tree model $\mathcal{I}_i$ by 'cutting' subtrees whose roots are connected from a_i via functional roles in F_i. With each set F_i, we associate a set Ω_{F_i} which is defined by

$$\Omega_{F_i} \coloneqq \{j\omega_{a_i} \in \Delta^{\mathcal{I}_i} \mid \exists \sigma \in F_i\colon (\epsilon_{a_i}, j_{a_i}) \in \sigma^{\mathcal{I}_i}, j \in \{1, \ldots, k\}, \omega \in \{1, \ldots, k\}^*\}$$

It is obvious that Ω_{F_i} contains precisely those elements of $\Delta^{\mathcal{I}_i}$ that form subtrees of $\mathcal{I}_i$ whose roots are connected from $a_i^{\mathcal{I}_i} = \epsilon_{a_i}$ via some functional role in F_i. Now, for every $1 \le i \le n$, an interpretation $\mathcal{J}_i$ is defined below:

$$\begin{aligned} \Delta^{\mathcal{J}_i} &\coloneqq \Delta^{\mathcal{I}_i} \setminus \Omega_{F_i} \\ a_i^{\mathcal{J}_i} &\coloneqq a_i^{\mathcal{I}_i} = \epsilon_{a_i} \\ A^{\mathcal{J}_i} &\coloneqq A^{\mathcal{I}_i} \cap \Delta^{\mathcal{J}_i} \text{ for each concept name } A \\ r^{\mathcal{J}_i} &\coloneqq r^{\mathcal{I}_i} \cap (\Delta^{\mathcal{J}_i} \times \Delta^{\mathcal{J}_i}) \text{ for each role name } r \end{aligned}$$

Note that for every $\mathcal{J}_i$, all elements of $\Delta^{\mathcal{I}_i}$ are retained in $\Delta^{\mathcal{J}_i}$ except the ones in Ω_{F_i}, in particular, the root ϵ_{a_i} is still retained as the root of $\mathcal{J}_i$. In addition, like in $\mathcal{I}_i$, for each $\mathcal{J}_i$, there is only one individual name occurring in $\mathcal{A}_{out}$, namely a_i, that is mapped to an element of $\Delta^{\mathcal{J}_i}$, i.e., the root ϵ_{a_i}. Thus, in the definition of $\mathcal{J}_i$ above, a_i only stands for that one particular individual name occurring in $\mathcal{A}_{out}$, and is not quantified over all individuals occurring in $\mathcal{A}_{out}$.

Next, we define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ by combining all interpretations $\mathcal{J}_i$, $i = 1, \ldots, n$ as follows:

$$
\begin{aligned}
\Delta^{\mathcal{I}} &\coloneqq \bigcup_{i=1}^{n} \Delta^{\mathcal{J}_i} \\
a_i^{\mathcal{I}} &\coloneqq a_i^{\mathcal{J}_i} = \epsilon_{a_i} \text{ for every individual name } a_i,\ i = 1, \ldots, n, \text{ occurring in } \mathcal{A}_{out} \\
A^{\mathcal{I}} &\coloneqq \bigcup_{i=1}^{n} A^{\mathcal{J}_i} \text{ for every concept name } A \\
r^{\mathcal{I}} &\coloneqq \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid r(a,b) \in \mathcal{A}_{out}\} \cup \bigcup_{i=1}^{n} r^{\mathcal{J}_i} \text{ for every role name } r
\end{aligned}
$$

We show that $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A}_{in})$ that does not satisfy $A_q(d)$. By definition of $A^{\mathcal{I}}$ and S_i for every $i = 1, \ldots, n$, Lemma 4.6, the fact that for every individual name a_i, $i = 1, \ldots, n$, occurring in $\mathcal{A}_{out}$, $a_i^{\mathcal{I}} = a_i^{\mathcal{J}_i} = a_i^{\mathcal{I}_i} \notin \Omega_{F_i}$, and since domains of $\mathcal{J}_{j_1}$ and $\mathcal{J}_{j_2}$ are disjoint for every $1 \leq j_1 < j_2 \leq n$, we have for every concept name A and individual name a_i, $i = 1, \ldots, n$,

$$A(a_i) \in \mathcal{A}_{out} \text{ iff } A \in S_i \text{ iff } a_i^{\mathcal{I}_i} \in A^{\mathcal{I}_i} \text{ iff } a_i^{\mathcal{I}} = a_i^{\mathcal{J}_i} \in A^{\mathcal{J}_i} \subseteq A^{\mathcal{I}}$$

Thus, we first obtain that $\mathcal{I}$ is a model of $\mathcal{A}_{out}$. Secondly, it is also a model of $\mathcal{A}_{in}$ because for every concept assertion $B(b) \in \mathcal{A}_{in}$, if b does not appear in $\mathcal{A}_{out}$ then $b \in P(a_i)$ for some $1 \leq i \leq n$ which implies $b^{\mathcal{I}} = a_i^{\mathcal{I}}$. Additionally, by some applications of **CR4** and/or **CR5**, we must have $B(a_i) \in \mathcal{A}_{out}$. This implies $b^{\mathcal{I}} = a_i^{\mathcal{I}} \in B^{\mathcal{I}}$. In addition, since we assume $d = a_1$, $d^{\mathcal{I}} = a_1^{\mathcal{I}}$ and $A_q(d) \in \mathcal{A}_{out}$ iff $A_q(a_1) \in \mathcal{A}_{out}$. Thus, if $A_q(a_1) \notin \mathcal{A}_{out}$ then $A_q \notin S_1$, which implies $d^{\mathcal{I}} = a_1^{\mathcal{I}} \notin A_q^{\mathcal{I}}$, i.e., $\mathcal{I}$ does not satisfy $A_q(d)$.

It now remains to show that $\mathcal{I}$ is a model of $\mathcal{T}$. To prove this, we distinguish cases based on possible forms of GCIs in $\mathcal{T}$ that is already normalized. In the following, we assume A, A_1, A_2 and B are concept names or $\top$, and r is a role name. Additionally, we use the fact that for every $1 \leq i \leq n$, $\mathcal{I}_{S_i}$ is a model of $\mathcal{T}$.

- $\top \sqsubseteq (\leq 1\ r)$. Let $v \in \Delta^{\mathcal{I}}$. If there is no $v' \in \Delta^{\mathcal{I}}$ such that $(v, v') \in r^{\mathcal{I}}$ then we are done. Thus, suppose there is at least one r-filler $v' \in \Delta^{\mathcal{I}}$ of v, i.e., $(v, v') \in r^{\mathcal{I}}$. We distinguish two cases based on whether or not $v \in \{a_1, \ldots, a_n\}$.

  - ⋆ $v \notin \{a_1, \ldots, a_n\}$. Then $\{v, v'\} \subseteq \Delta^{\mathcal{J}_i} \backslash \{a_i\} \subseteq \Delta^{\mathcal{I}_i} \backslash (\{a_i\} \cup \Omega_{F_i})$ for some i, $1 \leq i \leq n$. Hence, $v \in (\leq 1\ r)^{\mathcal{I}}$ is a consequence of the fact that $\mathcal{I}_i$ is a model of $\mathcal{T}$.
  - ⋆ $v = a_i^{\mathcal{I}}$ for some i, $1 \leq i \leq n$. First, if $r(a_i, b) \notin \mathcal{A}_{out}$ for any individual name b, then $v' \in \Delta^{\mathcal{J}_i}$ and $(v, v') \in r^{\mathcal{J}_i} \subseteq r^{\mathcal{I}_i}$. Hence, like above, $v \in (\leq 1\ r)^{\mathcal{I}}$ is a consequence of the fact that $\mathcal{I}_i$ is a model of $\mathcal{T}$. Second, if $r(a_i, b) \in \mathcal{A}_{out}$ for some individual name b occurring $\mathcal{A}_{out}$, then we show that $v' = b^{\mathcal{I}}$ and it is the only r-filler of v. Note that since **CR4** and **CR5** cannot be applied anymore to $\mathcal{A}_{out}$, we have that,

in fact, b is the only individual name occurring in $\mathcal{A}_{out}$ such that $r(a_i, b) \in \mathcal{A}_{out}$. It remains to show that $v' = b^{\mathcal{I}}$. Notice that $r \in F_i$. Moreover, it must be the case that $v' \notin \Delta^{\mathcal{I}_i}$, since otherwise, we would have $v' \in \Omega_{F_i}$ implying $v' \notin \Delta^{\mathcal{I}}$ which is a contradiction. Hence, by definition of $r^{\mathcal{I}}$, we obtain that $v' = b^{\mathcal{I}}$ from which it follows that $v \in (\leq 1\ r)^{\mathcal{I}}$.

- $\top \sqsubseteq (\leq 1\ r^-)$. Similar to the previous case.

- $A \sqsubseteq B$. Let $v \in \Delta^{\mathcal{I}}$ be such that $v \in A^{\mathcal{I}}$. By definition of $\mathcal{I}$, we have $v \in \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}$ and additionally, $v \in A^{\mathcal{J}_i} \subseteq A^{\mathcal{I}_i}$ for some i, $1 \leq i \leq n$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, $v \in B^{\mathcal{I}_i}$. Moreover, because B is also a concept name and it is obvious that $v \notin \Omega_{F_i}$, we obtain $v \in B^{\mathcal{J}_i} \subseteq B^{\mathcal{I}}$ by definition of $\mathcal{J}_i$ and $\mathcal{I}$. We thus conclude that $\mathcal{I}$ satisfies GCIs of the form $A \sqsubseteq B$.

- $A_1 \sqcap A_2 \sqsubseteq B$. Similar to the previous case.

- $A \sqsubseteq \exists r.B$ such that $\top \sqsubseteq (\leq 1\ r) \notin \mathcal{T}$ (r is not functional). Let $v \in \Delta^{\mathcal{I}}$ be such that $v \in A^{\mathcal{I}}$. By definition of $\mathcal{I}$, we have $v \in \Delta^{\mathcal{J}_i}$ and $v \in A^{\mathcal{J}_i} \subseteq A^{\mathcal{I}_i}$ for some i, $1 \leq i \leq n$. Then $v \in (\exists r.B)^{\mathcal{I}_i}$ because $\mathcal{I}_i$ is a model of $\mathcal{T}$. Thus, there is a $v' \in \Delta^{\mathcal{I}_i}$ such that $(v, v') \in r^{\mathcal{I}_i}$ and $v' \in B^{\mathcal{I}_i}$. If $v \neq \epsilon_{a_i}$, then obviously $v' \notin \Omega_{F_i}$. Otherwise, $v' \notin \Omega_{F_i}$ is due to $\top \sqsubseteq (\leq 1\ r) \notin \mathcal{T}$, i.e., $r \notin F_i$. In either case, we obtain $(v, v') \in r^{\mathcal{J}_i}$ and $v' \in B^{\mathcal{J}_i}$ yielding $v \in (\exists r.B)^{\mathcal{J}_i} \subseteq (\exists r.B)^{\mathcal{I}}$. It thus follows that $\mathcal{I}$ satisfies GCIs of the form $A \sqsubseteq \exists r.B$ such that $\top \sqsubseteq (\leq 1\ r) \notin \mathcal{T}$.

- $A \sqsubseteq \exists r^-.B$ such that $\top \sqsubseteq (\leq 1\ r^-) \notin \mathcal{T}$ (r^- is not functional). Similar to the previous case.

- $A \sqsubseteq \exists r.B$ such that $\top \sqsubseteq (\leq 1\ r) \in \mathcal{T}$ (r is functional). Let $v \in \Delta^{\mathcal{I}}$ be such that $v \in A^{\mathcal{I}}$. Thus, $v \in \Delta^{\mathcal{J}_i}$ and $v \in A^{\mathcal{J}_i} \subseteq A^{\mathcal{I}_i}$ for some i, $1 \leq i \leq n$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, it follows that $v \in (\exists r.B)^{\mathcal{I}_i}$ and $v \in (\leq 1\ r)^{\mathcal{I}_i}$. By the semantics, there is a unique $v' \in \Delta^{\mathcal{I}_i}$ such that $(v, v') \in r^{\mathcal{I}_i}$ and $v' \in B^{\mathcal{I}_i}$. We distinguish two cases.

  ⋆ $v \neq \epsilon_{a_i}$ or $v = \epsilon_{a_i}$ but there is no assertion $r(a_i, b) \in \mathcal{A}_{out}$ for some individual b in $\mathcal{A}_{out}$. Then $v' \notin \Omega_{F_i}$, i.e., $v' \in \Delta^{\mathcal{J}_i}$. This yields $(v, v') \in r^{\mathcal{J}_i}$ and $v' \in B^{\mathcal{J}_i}$, i.e., $v \in (\exists r.B)^{\mathcal{J}_i} \subseteq (\exists r.B)^{\mathcal{I}}$.

  ⋆ $v = \epsilon_{a_i}$ and there is an assertion $r(a_i, b) \in \mathcal{A}_{out}$ for some individual b in $\mathcal{A}_{out}$. Then $v' \in \Omega_{F_i}$ because $r \in F_i$. We thus have $v' \notin \Delta^{\mathcal{J}_i}$ yielding $v \notin (\exists r.B)^{\mathcal{J}_i}$. But after termination, **CR6** is no longer applicable to a_i and we obtain $B(b) \in \mathcal{A}_{out}$. Since $\mathcal{I}$ is a model of $\mathcal{A}_{out}$, we have $(a_i, b) \in r^{\mathcal{I}}$ and $b \in B^{\mathcal{I}}$ where $a_i^{\mathcal{I}} = a_i^{\mathcal{J}_i} = \epsilon_{a_i} = v$. Hence, we conclude $v \in (\exists r.B)^{\mathcal{I}}$.

  In either case, $v \in A^{\mathcal{I}}$ implies $v \in (\exists r.B)^{\mathcal{I}}$ for a functional r. We thus obtain that $\mathcal{I}$ satisfies $A \sqsubseteq \exists r.B$ such that r is functional.

- $A \sqsubseteq \exists r^-.B$ such that $\top \sqsubseteq (\leq 1\ r^-) \in \mathcal{T}$ (r^- is functional). Similar to the previous case using **CR7**.

- $\exists r.A \sqsubseteq B$. Let $v \in \Delta^{\mathcal{I}}$ such that $v \in (\exists r.A)^{\mathcal{I}}$. Note that $v \in \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}$ for some i, $1 \leq i \leq n$. We distinguish two cases.

  ⋆ $v \in (\exists r.A)^{\mathcal{J}_i}$. Then there is a $v' \in \Delta^{\mathcal{J}_i}$ such that $(v, v') \in r^{\mathcal{J}_i} \subseteq r^{\mathcal{I}_i}$ and $v' \in A^{\mathcal{J}_i} \subseteq A^{\mathcal{I}_i}$. Thus, $v \in (\exists r.A)^{\mathcal{I}_i}$. Since $\mathcal{I}_i$ is a model of $\mathcal{T}$, $v \in B^{\mathcal{I}_i}$. By definition of $\mathcal{J}_i$ and $\mathcal{I}$, we have $v \in B^{\mathcal{J}_i} \subseteq B^{\mathcal{I}}$ as B is a concept name and $v \notin \Omega_{F_i}$.

  ⋆ $v \notin (\exists r.A)^{\mathcal{J}_i}$. Then there is no $v' \in \Delta^{\mathcal{J}_i}$ such that $(v, v') \in r^{\mathcal{J}_i}$ and $v' \in A^{\mathcal{J}_i}$. Since such a $v' \in \Delta^{\mathcal{I}}$ such that $v' \in r^{\mathcal{I}}$ and $v' \in A^{\mathcal{I}}$ must exist, by definition of $r^{\mathcal{I}}$, $v = \epsilon_{a_i} = a_i^{\mathcal{I}}$ and $v' = a_j^{\mathcal{I}}$ for some individual name a_j, $1 \leq j \leq n$, such that $r(a_i, a_j) \in \mathcal{A}_{out}$. Note that $a_j^{\mathcal{J}_j} = \epsilon_{a_j}$ and $\epsilon_{a_j} \in A^{\mathcal{J}_j} \subseteq A^{\mathcal{I}_j}$. Because $\mathcal{I}_j$ is obtained from Lemma 4.6, it holds that $A \in S_j$, i.e., $A(a_j) \in \mathcal{A}_{out}$. Thus, precondition of **CR1** is fulfilled. But after termination, **CR1** is no longer applicable. This means $B(a_i) \in \mathcal{A}_{out}$ holds. Since we have $\mathcal{I}$ is a model of $\mathcal{A}_{out}$, we conclude $a_i^{\mathcal{I}} \in B^{\mathcal{I}}$, i.e., $v \in B^{\mathcal{I}}$.

  In both cases, we have $v \in (\exists r.A)^{\mathcal{I}}$ implies $v \in B^{\mathcal{I}}$. Hence, $\mathcal{I}$ satisfies GCIs of the form $\exists r.A \sqsubseteq B$. Observe that this is regardless whether r is functional or not.

- $\exists r^{-}.A \sqsubseteq B$. Similar to the previous case using **CR2**.

As every form of GCIs is satisfied by $\mathcal{I}$, we conclude that $\mathcal{I}$ is indeed a model of $\mathcal{T}$ and a countermodel of $(\mathcal{T}, \mathcal{A}_{in}) \models A_q(d)$. ◇

The last proposition finishes our analysis of the data complexity of instance checking for $\mathcal{ELI}^f$. Finally, we sum up everything in the following theorem.

Theorem 4.14
Data complexity of instance checking for $\mathcal{ELI}^f$ w.r.t. general TBoxes is polynomial. ◇

Chapter 5

Conclusion

The subject of this work was to map out the data complexity of the instance checking problem for extensions of $\mathcal{EL}$. Instance checking is the simplest form of query answering over $\mathcal{EL}$ knowledge bases. It is the problem of deciding whether all axioms in the input TBox and all assertions in the input ABox logically imply that the input individual belongs to the input concept. In the literature, the complexity of this problem is usually measured in the size of the whole input which consists of a TBox, an ABox, a query concept and an individual name, hence named combined complexity. However, we have argued in Chapter 2 that there is another relevant complexity measure, namely data complexity, which is measured only in the size of the input ABox. Data complexity is relevant because in many applications, the size of data, i.e., the number of assertions in the input ABox is much larger than the size of size of terminologies involved.

Baader *et al.* [2005a] have shown that adding several common description logic constructors to $\mathcal{EL}$ yields to several extensions of $\mathcal{EL}$ for which the subsumption problem w.r.t. general TBoxes is ExpTime-complete. In Chapter 3, we have identified among these extensions of $\mathcal{EL}$, the ones for which data complexity of instance checking w.r.t. general TBoxes is coNP-hard (and even coNP-complete). These include extensions of $\mathcal{EL}$ with negation, disjunction, value restriction, number restriction and some role constructors including role complement, role union and transitive closures, and for most of them, coNP-hardness were obtained even without the presence of TBoxes. In order to derive the coNP-hardness results, we adapted the technique described by Schaerf [1993] using a polynomial reduction from a variant of SAT, called 2+2-SAT, which was employed to show the coNP-hardness regarding data complexity of instance checking in the DL $\mathcal{ALE}$. Whereas, for the coNP upper bound, we refer to [Hustadt *et al.*, 2005] which established coNP-completeness regarding data complexity of instance checking for DLs which provide at least the constructors from the DL $\mathcal{ALC}$ and at most the constructors from the DL $\mathcal{SHIQ}$.

In Chapter 4, we have identified one extension of $\mathcal{EL}$ for which data complexity of instance checking is polynomial. This language, named $\mathcal{ELI}^f$, is obtained by adding inverse roles and functionality to $\mathcal{EL}$. This result is notable, since Baader *et al.* [2005a,b] have shown that by merely adding one of these constructors yields to ExpTime-completeness of subsumption w.r.t. general TBoxes. The tractability result was proved by providing an algorithm that decides instance checking w.r.t. general TBoxes and runs in time polynomial in the size of input ABox.

We summarize all results about data complexity derived in this thesis in Table 5.1.

Extensions of $\mathcal{EL}$	Data complexity of inst. checking with simple ABoxes		
	without TBoxes	w.r.t. acyclic TBoxes	w.r.t. general TBoxes
$\mathcal{EL}^{(\neg)}$, $\mathcal{EL}^{\neg}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{ELU}$	in P	coNP-complete	coNP-complete
$\mathcal{EL}^{\forall r.\bot}$, $\mathcal{EL}^{\forall}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\leq k_1, \geq k_2}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\leq k}$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{kf}$, $k \geq 2$	coNP-complete	coNP-complete	coNP-complete
$\mathcal{EL}^{\geq 2}$	in P	coNP-complete	coNP-complete
$\mathcal{EL}^{\geq k}$, $k \geq 3$	in P	in coNP, hardness still open	coNP-complete
$\mathcal{EL}^{R\neg}$	coNP-hard	coNP-hard	coNP-hard
$\mathcal{EL}^{\cup}$, $\mathcal{EL}^{*}$	in P	coNP-hard	coNP-hard
$\mathcal{ELI}^{f}$	in P	in P	in P

Table 5.1: Data complexity of instance checking for various extensions of $\mathcal{EL}$

In the table, k, k_1 and k_2 are fixed nonnegative integers. Note that $\mathcal{ELI}$ and $\mathcal{EL}^{1f}$ are sublanguages of $\mathcal{ELI}^f$, and thus share the tractability result of $\mathcal{ELI}^f$. Missing results in the table are the coNP-hardness regarding data complexity of instance checking w.r.t. acyclic TBoxes for the DL $\mathcal{EL}^{\geq k}$, $k \geq 3$, as well as some coNP upper bound results of data complexity of instance checking for $\mathcal{EL}^{R\neg}$, $\mathcal{EL}^{\cup}$, and $\mathcal{EL}^{*}$.

From the table, we would like to point out some interesting findings in this thesis. First, the tractability boundary between combined complexity and data complexity apparently does not coincide due to the data complexity result for $\mathcal{ELI}^f$ for which combined complexity is intractable. Second, a slight syntactic difference between local and global (1)-functionality has turned out to yield a computational "cliff" in terms of data complexity. And finally, our findings showed that in most cases, there is no difference between adding either general TBoxes or acyclic TBoxes. The difference, if it does exist, turned out to be between with and without TBox. This can be seen on results for $\mathcal{ELU}$, $\mathcal{EL}^{\geq 2}$, $\mathcal{EL}^{\cup}$, and $\mathcal{EL}^{*}$. A possible exception is for $\mathcal{EL}^{\geq k}$, $k \geq 3$, in which data complexity of instance checking w.r.t. acyclic TBoxes is not completely characterized.

There are some directions for future work that can be considered. One possible future work is to investigate the aforementioned missing results and fill up Table 5.1 to make a more complete general picture. Another possible future work could be extending $\mathcal{ELI}^f$ with other constructors from the DL $\mathcal{EL}^{++}$ [Baader *et al.*, 2005a,b]. Since $\mathcal{EL}^{++}$ is known to be tractable even for combined complexity, it is natural to conjencture that by adding inverse roles and functionality, tractability regarding the data complexity can still be retained. In addition, one could also generalize the reasoning problem of instance checking into more complex form of conjunctive query answering. In this direction, it would be interesting to investigate whether tractability of instance checking regarding data complexity can also be retained. Finally, since the algorithm which was described in Chapter 4 was basically provided as a means to prove that instance checking w.r.t. general TBoxes for $\mathcal{ELI}^f$ is tractable regarding data complexity, as a future work, one could also think of a real optimized implementation of the algorithm that behaves well in practice.

Bibliography

F. Baader (1996). Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219.

F. Baader (2003a). The instance problem and the most specific concept in the description logic EL w.r.t. terminological cycles with descriptive semantics. In *Proceedings of the 26th Annual German Conference on Artificial Intelligence (KI 2003)*, vol. 2821 of *Lecture Notes of Artificial Intelligence*, pp. 64–78. Springer-Verlag, Hamburg, Germany.

F. Baader (2003b). Terminological cycles in a description logic with existential restrictions. In G. Gottlob & T. Walsh (eds.), *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9–15, 2003*, pp. 325–330. Morgan Kaufmann.

F. Baader, S. Brandt & C. Lutz (2005a). Pushing the EL envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05).* Morgan-Kaufmann Publishers, Edinburgh, UK.

F. Baader, S. Brandt & C. Lutz (2005b). Pushing the EL envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany. See http://lat.inf.tu-dresden.de/research/reports.html.

F. Baader, D. Calvanese, D. McGuinness, D. Nardi & P. F. Patel-Schneider (eds.) (2003). *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.

F. Baader & W. Nutt (2003). Basic description logics. In Baader *et al.* [2003], chap. 2, pp. 43–95.

A. Borgida, M. Lenzerini & R. Rosati (2003). Description logics for databases. In Baader *et al.* [2003], chap. 16, pp. 462–484.

R. J. Brachman & H. J. Levesque (1984). The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI'84)*, pp. 34–37.

R. J. Brachman & H. J. Levesque (eds.) (1985). *Readings in Knowledge Representation.* Morgan Kaufmann, San Mateo, CA.

R. J. Brachman & J. G. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216.

S. Brandt (2004a). On subsumption and instance problem in ELH w.r.t. general TBoxes. In V. Haarslev & R. Möller (eds.), *Description Logics*, vol. 104 of *CEUR Workshop Proceedings*. CEUR-WS.org.

S. Brandt (2004b). Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras & L. Saitta (eds.), *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pp. 298–302. IOS Press.

M. Buchheit, F. M. Donini & A. Schaerf (1993). Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, (1).

D. Calvanese & G. De Giacomo (2003). Expressive description logics. In Baader *et al.* [2003], chap. 5, pp. 178–218.

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini & R. Rosati (2006). Data complexity of query answering in description logics. In *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pp. 260–270.

R. Cote, D. Rothwell, J. Palotay, R. Beckett & L. Brochu (1993). The systematized nomenclature of human and veterinary medicine. Tech. rep., SNOMED International, Northfield, IL.

F. M. Donini (2003). Complexity of reasoning. In Baader *et al.* [2003], chap. 3, pp. 96–136.

F. M. Donini, M. Lenzerini, D. Nardi & W. Nutt (1991). The complexity of concept languages. In J. Allen, R. Fikes & E. Sandewall (eds.), *Proceedings of the 2nd International Conference on the Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 151–162. Morgan Kaufman.

F. M. Donini, M. Lenzerini, D. Nardi & A. Schaerf (1994). Deduction in concept languages: from subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452.

P. J. Hayes (1979). The logic of frames. In D. Metzing (ed.), *Frame Conceptions and Text Understanding*, pp. 46–61. Walter de Gruyter and Co., Berlin. Republished in Brachman & Levesque [1985].

J. Heflin & J. Hendler (2001). A portrait of the semantic web in action. *IEEE Intelligent Systems*, 16(2):54–59.

U. Hustadt, B. Motik & U. Sattler (2005). Data complexity of reasoning in very expressive description logics. In L. P. Kaelbling & A. Saffiotti (eds.), *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 466–471. Professional Book Center. URL `http://www.ijcai.org/papers/0326.pdf`.

F. Lehmann (ed.) (1992). *Semantic Networks in Artificial Intelligence.* Pergamon Press, Oxford, United Kingdom.

M. Lenzerini (2002). Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, pp. 233–246.

M. Minsky (1974). A framework for representing knowledge. Tech. Rep. 306, Artificial Intelligence Laboratory, MIT. Republished in Brachman & Levesque [1985].

D. Nardi & R. J. Brachman (2003). An introduction to description logics. In Baader *et al.* [2003], chap. 1, pp. 1–40.

B. Nebel (1988). Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383.

B. Nebel (1990a). *Reasoning and Revision in Hybrid Representation Systems.* No. 422 in Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin, Germany.

B. Nebel (1990b). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249.

C. H. Papadimitriou (1994). *Computational Complexity.* Addison Wesley Publishing Company.

M. R. Quillian (1968). Semantic memory. In M. Minsky (ed.), *Semantic Information Processing*, pp. 227–270. MIT Press, Cambridge, Massachusetts.

A. Rector & I. Horrocks (1997). Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97).* Stanford, CA.

R. Reiter (1984). Towards a logical reconstruction of relational database theory. In M. Brodie, J. Mylopoulos & J. Schmidt (eds.), *On Conceptual Modelling.* Springer-Verlag.

U. Sattler, D. Calvanese & R. Molitor (2003). Relationship with other formalisms. In Baader *et al.* [2003], chap. 4, pp. 137–177.

A. Schaerf (1993). On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, 2(3):265–278.

A. Schaerf (1994). *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues.* Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza".

M. Schmidt-Schauß & G. Smolka (1991). Attributive concept descriptive with complements. *Artificial Intelligence*, 48(1):1–26.

K. A. Spackman (2001). Normal forms for description logic expressions of clinical concepts in SNOMED-RT. *Journal of the American Medical Informatics Association*. Symposium Supplement.

The Gene Ontology Consortium (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*.

S. Tobies (2001). *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. Ph.D. thesis, Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen. See http://lat.inf.tu-dresden.de/research/phd/index.html.

M. Vardi (1986). Querying logical database. *Journal of Computer and System Science*, 33:142–160.