Learning Description Logic Knowledge Bases from Data Using Methods from Formal Concept Analysis

Dissertation

zur Erlangung des akademischen Grades Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Math. Felix Distel
geboren am 25. März 1981 in Geislingen an der Steige

verteidigt am 27. April 2011

Gutachter:
Prof. Dr.-Ing. Franz Baader,
Technische Universität Dresden
Prof. Dr. rer. nat. Gerd Stumme,
Universität Kassel

Dresden, im Juni 2011

Contents

List of Algorithms

List of Figures

1 Introduction

An important field within Artificial Intelligence research is Knowledge Representation. Description Logics (DL) are a succesful family of formalisms for Knowledge Representation [BCM$^+$03]. They have gained recognition in various fields outside logics during the last decades. A DL knowledge base is essentially a collection of axioms. Each axiom can either describe knowledge about categories or classes of entities, so-called *terminological knowledge*, or knowledge about concrete entities, so-called *assertional knowledge*. These axioms are expressed using a formally defined syntax and semantics. For example, to express the terminological knowledge that "A father is male and has a child." one could write in DL Syntax

$$\mathsf{Father} \sqsubseteq \exists \mathsf{Male} \sqcap \mathsf{hasChild}.\top.$$

While the syntax and semantics of these axioms may be intuitive to logicians, they can be daunting to experts from other fields. In order to make DL accessible to experts from other fields it is necessary to develop procedures that assist knowledge engineers during the design process.

We develop formalisms that automatically generate axioms. These axioms are then presented to the knowledge engineer who can decide whether they should be added to the knowledge base, the rationale being that it is easier to understand and accept a GCI than to generate it by hand. Of course, these axioms should not be generated randomly. First, they should not contradict existing knowledge (for example because a known individual serves as a counterexample). Second, they should add as much information as possible, ideally culminating in a *complete* knowledge base. In this thesis we provide answers to the following questions.

- When can a knowledge base be considered *complete*, and can complete knowledge bases even exist?

- How can a complete knowledge base be reached in interaction with a (potentially human) expert?

Our formalisms to reach completeness follow the pattern of *Attribute Exploration*, a technique from Formal Concept Analysis. Formal Concept Analysis (FCA) is a field from discrete mathematics that is not directly related to DL, although similarities exist. Before we can use it for our purposes we need to answer the third question:

- What is a good framework to bridge the gap between Formal Concept Analysis and Description Logics?

We discuss each of these problems in more detail after a brief introduction to DL and FCA.

1.1 Description Logics

Early knowledge representation formalisms such as *semantic networks* [Sow91] and *frames* [Min81] were motivated from applications, mostly within the field of linguistics. The roots of Description Logics lie within these fields. Semantic Networks are a graphical representation of conceptual knowledge. They represent knowledge in the form of a labelled, directed graph, where nodes represent either individuals or classes of individuals and edges represent relationships between nodes. There are numerous classes of semantic networks, e. g. *definitional networks* with an emphasis on *is-a*-relations that define a hierarchy over individuals or classes, or *learning networks* with a non-monotonic flavour [Sow92]. Frames, on the other hand, ressemble the class structures known from object oriented programming. Concepts are represented as frames, which possess a set of super-concepts and a set of *slots*. Relationships between concepts are realized using slots, which can link to other frames. Both semantic networks and frames lack a formally defined semantics. The meaning that is associated to a network or a frame largely depends either on the intuition of its author or the specific implementation of the reasoning system. Knowledge representation systems with a logical underpinning arose from the need for a more predictable and realiable form of knowledge representation [BS85]. These systems eventually evolved into Description Logics.

Even though the early Description Logics were relatively small fragments of First Order Logics, first intractability results were obtained already in the 1980s [BL84, Neb88]. From then on, most research was devoted to the development and optimization of reasoning algorithms that, despite the exponential (or worse) worst-time complexity, behave well in practice [HST00, HS04]. Meanwhile, a range of new constructors has been added to increase expressivity – eventually leading to the recommendation by the W3C of the *Web Ontology Language OWL* which is based on the expressive DL language $\mathcal{SHOIN}$ [HPSvH03]. Efficient algorithms were implemented in reasoning systems for expressive DLs such as FaCT [Hor98, TH06] and RACER [HM01].

It was not until the early 2000s that light-weight DLs returned to the focus of researchers. For the Description Logic $\mathcal{EL}$ which allows for conjunction and existential restrictions it was shown that standard reasoning services are tractable for cyclic and acyclic TBoxes [Baa03b] as well as for general TBoxes [Bra04]. Later, more expressive extensions of $\mathcal{EL}$ have been designed, which maintain tractability [BBL05a, BBL08]. The expressivity of $\mathcal{EL}$ or variants thereof is sufficient in many applications, in particular in medicine or biology. Examples are the large biomedical ontology SNOMED [SCC97], the Gene Ontology [ABB+00] and large parts of the GALEN medical knowledge base [RH97]. An efficient reasoning system for the $\mathcal{EL}$ family of Description Logics is available under the name CEL [BLS06, MS09].

New reasoning services that assist the design process of Description Logic ontologies have been a second major research direction in the 2000s. We provide an overview of these in Section 1.1.2 and present some of them in more detail in Section 2.3.

1.1.1 Describing Knowledge and Standard Reasoning

Classically, DL knowledge bases, also called ontologies, consist of two components: the *TBox* containing terminological knowledge, and the *ABox* containing assertional knowledge. The terminological knowledge contained in the TBox is expressed by means of *concept descriptions*. Each DL provides a number of *concept constructors* which, together with *concept names* and *role names*, can be used to generate concept descriptions. $\exists \mathsf{hasChild}.\top$, which has been used in the above example, is a concept description that uses $\exists$ and $\top$ as concept constructors and

hasChild as a role name. It roughly translates to "Someone who has a child" in natural language. New concept names can be assigned to concept descriptions using *concept definitions*. One might state

$$\mathsf{Parent} \equiv \exists \mathsf{hasChild}.\top$$

to express that a parent is someone who has a child. Depending on the type of TBox it may also be possible to express that a concept description is more specific than another using *general concept inclusions (GCIs)*:

$$\mathsf{Father} \sqsubseteq \exists \mathsf{hasChild}.\top.$$

Concept definitions and GCIs are stored in the TBox.

Knowledge bases typically contain knowledge about individuals in addition to terminological knowledge. DLs allow to express that an individual belongs to a concept, or that it is linked to another individual by a role name. For example, we can state that Homer is a Father, and that Bart is Lisa's brother using the following assertions:

$$\mathsf{Father}(\mathsf{Homer})$$
$$\mathsf{hasBrother}(\mathsf{Lisa}, \mathsf{Bart}).$$

Assertions about individuals are stored in the ABox.

In contrast to earlier KR-formalisms, DL systems have a formally defined semantics. This semantics is based on the notion of *models* and *interpretations*. An interpretation consists of a finite set, called *domain*, and a mapping that associates individuals with elements of the domain, concept names with subsets of the domain and role names with binary relations over the domain. A model of a knowledge base is an interpretation that does not violate the statements from the TBox or the ABox.

DL knowledge bases use an open-world semantics, i. e. absence of information is interpreted as lack of knowledge, not as negation of information. Some of our results require closed-world knowledge about individuals. In this case we use models as a closed-world representation of individuals in a DL setting.

The formal model-theoretic semantics enables us to reason about knowledge. Using reasoning services it is possible to obtain implicit knowledge from explicit knowledge. The explicit knowledge that every

Father has a child, and that Homer is a Father yields the implicit knowledge that Homer has a child. The combination of a knowledge base and a reasoning component is called a *DL system*. Most DL systems support at least a number of *standard reasoning services*.

Among these is *subsumption*-reasoning, i. e. deciding whether one concept is more specific than another. If the system computes all subsumption relationships for concepts from the TBox then we also speak of the *classification* of an ontology. *Instance checking* denotes the task of verifying whether an individual belongs to a concept. The above inference that Homer belongs to the concept $\exists \mathsf{hasChild}.\top$ is an example for instance checking. Sometimes it may occur, that due to design errors a knowledge base contains contradictory information. This can be detected using *consistency checking*. A less severe consequence of design errors occurs when a concept becomes unsatisfiable, i. e. no individuals can belong to this concept. *Satisfiability checking* can be used to detect such concepts.

1.1.2 Reasoning Services to Support Ontology Design and Maintenance

Standard reasoning services have been designed to infer implicit knowledge from knowledge that is explicitly present in an existing ontology. Non-standard reasoning services, by contrast, have been designed with the design and maintenance of ontologies in mind.

Axiom Pinpointing Since the design of knowledge bases is an error-prone process, a common maintenance task is the removal of faulty information from a knowledge base. This is straightforward if the faulty knowledge is explicit knowledge: one can simply remove or modify the faulty axiom. If it is implicit knowledge it is less clear which axioms need to be removed or modified. Whenever an unwanted consequence is discovered *axiom pinpointing* can be used to identify the axioms that are responsible for the error [BH95, SC03, PSK05, Peñ09].

Ontology Import Reusing knowledge from existing ontologies can tremendously speed up the ontology engineering process. Typically, an ontology engineer wants to be sure to import all axioms from the exist-

ing ontology that are relevant in the domain of the new ontology. On the other hand, it is important to keep ontologies small, not least in consideration of the complexity of reasoning. *Module Extraction* can be used to identify a subset of the existing ontology that is small, yet still captures the meaning of the imported concepts. A second problem can occur when parts of an ontology are imported: The language of the existing ontology is not necessarily the same as the language of the new ontology. In this setting *concept approximation* can be used to determine for a concept description from the existing DL the closest concept description in the new DL [BKT02].

Bottom-Up Construction Usually, ontologies are constructed in a top-down fashion, where more general concepts are defined first, and then gradually specified. The *bottom-up approach* for ontology design provides an alternative [BT01]. Instead of starting with concept descriptions the modeler starts with the individuals. An engineer who wants to describe a concept selects a set of individuals that belong to this concept. The DL system then automatically generates a concept description which describes these individuals. This concept description is then presented to the modeler who can use or modify it. Two reasoning services are used to obtain the suggested concept description: *most specific concepts* and *least common subsumers*. For each of the selected individuals the DL system computes the most specific concept of which it is an instance. The obtained concept descriptions are then generalized into one concept description by computing the least common subsumer. The least common subsumer is the least general concept description that generalizes the concepts it is applied to [BKM99, Baa03a, Tur07].

Knowledge Base Completion Knowledge Base Completion is a tool to help knowledge engineers to ensure that their knowledge base contains all the relevant information about the domain. Its objective is to assist the engineer in adding information until

- all relationships between concepts that are relevant in the domain are captured by the TBox, and
- all kinds of instances are represented by an individual in the ABox.

In this thesis we present a knowledge base completion formalism. It can be viewed as an enhancement of the interactive method for knowledge base completion which has been proposed in [BGSS07, Ser07]. It follows the pattern of Attribute Exploration from FCA. We go into more detail on this approach after an introduction to FCA.

1.2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a branch of mathematical order theory, or more precisely a branch of lattice theory [Bir93] that has emerged during the 1980s. It is considered to be an applied branch of lattice theory, because of applications in data-mining, among others. It focusses on a special type of lattices that are obtained from binary relations, called *formal contexts*. It shares with DL the notions of concepts as describing a collection of objects with common properties. However, the two fields differ fundamentally in the way concepts are obtained. While the logic based concept descriptions in DL are usually generated and edited manually by a modeler, concepts in FCA are generated automatically from the formal context using *derivation operators*. FCA also possesses a tool to semi-automatically analyze dependencies between concepts called *Attribute Exploration*.

1.2.1 Foundations

Data in Formal Concept Analysis is represented as a *formal context* which consists of a set of *objects*, a set of *attributes* and a binary relation, the *incidence relation* that determines whether an object has an attribute. They are usually visualized as cross-tables like in Table 1.1. For example, the cross in the first row and second column of Table 1.1 implies that the object Homer has the attribute Male. The absence of a cross in the next column implies that Homer does not have the attribute Mother. In classical formal contexts it is not possible to express that it is unknown whether Homer is a Parent. From a DL viewpoint formal contexts have a *closed-world semantics*.

Derivation Operators map sets of attributes to sets of objects and vice versa. The derivation of a set of attributes A is defined as the set B of all objects that have all attributes from A. For example, in Table 1.1

Table 1.1: A Formal Context $\mathbb{K}_S$ about Two Families

	Female	Male	Mother	Father	Parent
Homer		×		×	×
Marge	×		×		×
Bart		×			
Lisa	×				
Kirk		×		×	×
Luann	×		×		×
Milhouse		×			

the derivation of the set $\{\mathsf{Male}, \mathsf{Father}\}$ is $\{\mathsf{Homer}, \mathsf{Kirk}\}$. Conversely, the derivation of a set of objects B is defined as the set A of all attributes that are shared by all objects from B. Derivation operators give rise to *formal concepts*: pairs consisting of a set of objects and a set of attributes, where the former is the derivation of the latter and vice versa. An example for a formal concept in Table 1.1 is the pair $(\{\mathsf{Marge}, \mathsf{Luann}\}, \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\})$. The first part of a formal concept is called the *extent*, and the second is called the *intent* of the concept. This notion of a concept appears to be fundamentally different from the notion of concepts from DL, but similarities exist nevertheless. In both cases, there is a description that describes, based on certain properties, which objects or individuals belong to the concept. In the case of FCA this is the concept intent $\{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\}$. In DL it could be a concept description such as $\mathsf{Female} \sqcap \mathsf{Mother} \sqcap \mathsf{Parent}$. The formal concepts of a formal context can be ordered, where set inclusion of the concept extends is used as the order relation. The resulting structure is a lattice like the one shown in Figure 1.1. Concept lattices are an important tool for visualizing the contents of a formal context.

1.2.2 Implications and Attribute Exploration

The concept lattice is one way to analyze a formal context, *implications* are another. In Table 1.1 all objects that have the attributes Female and

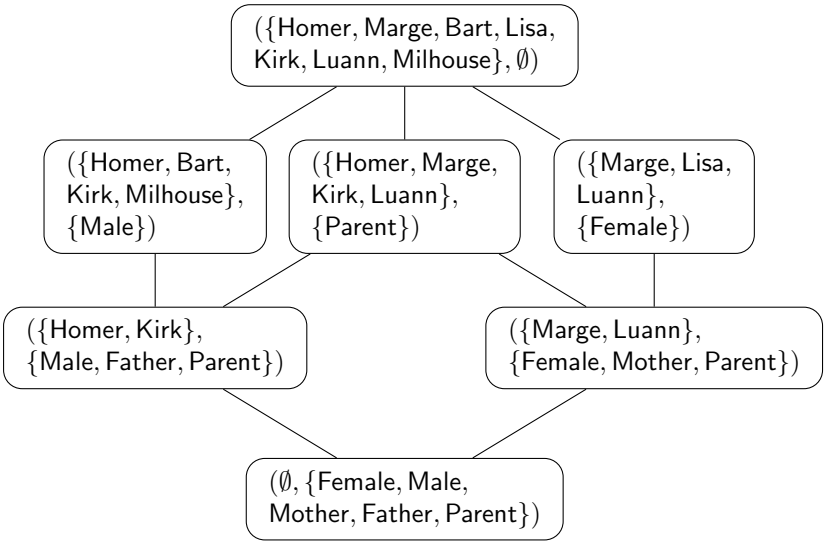

Figure 1.1: The Concept Lattice for Table 1.1

Parent also have the attribute Mother. A shorter way to write this is

$$\{\mathsf{Female}, \mathsf{Parent}\} \to \{\mathsf{Mother}\}.$$

This kind of dependency is called an *implication*. We have already mentioned parallels between sets of attributes in FCA and concept descriptions in DL. From this perspective an implication can be viewed as a dependency between concept descriptions, quite similar to a GCI in Description Logics. Unlike GCIs in DL, which need to be created manually, FCA provides several approaches to obtain sets of implications from a formal context. The most well-known is the *Duquenne-Guigues Base*. It is a minimal set of implications that hold in a given context, and from which all implications that hold in the context follow semantically. The Duquenne-Guigues Base can be computed if the context is fully known.

For the case where not all the objects from the context are known FCA provides the tool *Attribute Exploration*. In many applications the context is not explicitly given, but is known to a *domain expert*. Attribute Exploration is an efficient method to retrieve the expert's knowledge. It starts with an empty set of implications $\mathcal{L} = \emptyset$ and successively generates questions of the form "Do all objects with attributes $m_1, \ldots, m_k$ also have the attributes $m'_1, \ldots, m'_\ell$?". Upon seeing the question the expert has two choices. She can either answer "Yes", in which case the implication $\{m_1, \ldots, m_k\} \to \{m'_1, \ldots, m'_\ell\}$ is added to the set $\mathcal{L}$. If the expert answers "No", then she is asked to provide a counterexample, i. e. an object that does not satisfy the implication. This object is then added to the context which gradually becomes more similar to the full context. The questions are chosen in a smart way that guarantees that upon termination the set $\mathcal{L}$ is the Duquenne-Guigues-Base of the initially unknown full context. This ensures that the number of accepted implications is minimal, keeping the number of costly expert interactions low.

1.3 Existing Exploration Formalisms

We briefly introduce the two existing approaches that apply Attribute Exploration in a DL setting. A more detailed comparison between these approaches and the work from this thesis requires a deeper insight into

the technical parts. Therefore it will be provided in Chapter 8 about related work.

Completion Using OntoComp In [BGSS07] a first ontology completion formalism based on Attribute Exploration has been introduced. In aims at a well-defined, yet relatively weak notion of completeness. It starts with an existing ontology. As a first step the ontology engineer is asked to select a set M of interesting concepts that occur in the TBox. The algorithm then considers only GCIs of the type

$$\bigsqcap U \sqsubseteq \bigsqcap V,$$

where $U, V \subseteq M$ are subsets of M. It is assumed that the ontology engineer has complete knowledge about the domain of interest, and that this knowledge can be represented as a model of the ontology. An ontology is called *complete* if

- all GCIs of the above type that hold in the domain follow from the ontology, and
- for every GCI of the above type that does not hold in the domain there is a counterexample in the ontology.

The method used to achieve this is a version Attribute Exploration that is tailored to a DL setting. It follows the typical pattern, where hypotheses are successively presented to the expert, who may either accept them or refute them. If the hypothesis is accepted, then it is added to the TBox of the ontology. If the hypothesis is refuted, the expert is required to provide a counterexample, which is added to the ABox. The main contribution of [BGSS07] is a framework for dealing with the ontology's open-world knowledge in an Attribute Exploration setting. This is achieved by adapting the notion of a context to *partial contexts*. In contrast to older results about FCA with incomplete knowledge such as [Bur91, BH00] it is assumed that the expert has complete knowledge about the domain. This system has been improved further with respect to certain usability issues [Ser08, BS09]. An implementation is available as a plugin for the ontology editor *Protégé* under the name *OntoComp* [Ser09a].

Relational Exploration *Relational Exploration* [Rud06]. is perhaps the closest existing formalism to the formalisms presented in this thesis. Unlike the approach used for OntoComp, it uses a stronger notion of completeness where no restrictions on the types of GCIs are made. However, this notion of completeness is not entirely compatible with the usual semantics of DL (cf. 8.2.2 for more details). The DL that is used is called $\mathcal{FLE}$. Relational Exploration starts, not with an ontology, but with a closed-world representation of parts of the domain called *binary power context family*. Rudolph considers an infinite family of formal contexts where the attributes are DL concept descriptions of increasing role depth. The objects of these contexts and the incidence relation are obtained from the binary power context family. Subsequently, classical Attribute Exploration is performed on the resulting contexts, until a certain termination condition is reached. Rudolph shows that, for a finite binary power context family, this termination condition will always be satisfied eventually, and that the implication bases of the contexts considered up to that step contain enough information to decide, for any GCI between $\mathcal{FLE}$-concepts, whether this GCI holds in the domain.

In summary, we can say that there is an existing formalism that allows for open-world knowledge but can only deal with a weak notion of completeness, and there is an existing formalism that can deal with a strong notion of completeness but requires closed-world knowledge. One goal of this thesis is to obtain a formalism that combines the best of both worlds: open-world knowledge and a stronger notion of completeness. We do not use Rudolph's algorithms directly for our approach, since – apart from lacking the capacity to deal with open-world semantics – there are other issues such as the great increase in the number of attributes as role depth increases. We discuss these issues in Section 8.2.2.

1.4 Contributions

The main contributions of this thesis are

- an approach to bridge the gap between FCA and DL, called model-based most specific concepts,
- the proof that it is always possible to find a finite set of GCIs that is complete for a given $\mathcal{EL}^{\bot}$-model,

- Model Exploration, a formalism inspired by Attribute Exploration, where counterexamples are stored in a model, and
- ABox Exploration, an exploration formalism where counterexamples are stored in an ABox.

Throughout this work we use the lightweight DL $\mathcal{EL}^{\bot}$ and its extension $\mathcal{EL}^{\bot}_{\text{gfp}}$. Both logics have the advantage that efficient reasoning is possible. They are subsets of $\mathcal{EL}^{++}$, a widely recognized extension of $\mathcal{EL}$ [BBL05a].

Model-Based Most Specific Concepts In order to apply techniques from FCA to DL knowledge bases we first need a framework that bridges the gap between the two fields. We have already pointed out the similarities between concept descriptions in DL and sets of attributes in FCA: Both can describe sets of individuals or objects by specifying their properties. A similar connection can be made between DL models, ABoxes, and formal contexts. Each of them contains the information which individuals or objects have which properties. In this first part, we consider a strictly closed-world setting and therefore use only models and not ABoxes.

The connections between concept descriptions and sets of attributes on the one hand and formal contexts and models on the other have been exploited already in the first works on FCA and DL. Most authors have used an approach where models are transformed into contexts whose attributes are concept descriptions. In this work, we call such contexts *induced contexts*. Induced contexts are relatively easy to obtain, however, it is not immediately clear how one should choose the concept descriptions that will serve as attributes.

We pursue a different idea where we try to transport techniques from FCA to DL and not the other way around. Perhaps, the most basic operations in FCA are the derivation operators. One of them maps sets of attributes to sets of objects and the other maps sets of objects to sets of attributes. In DL, a suitable operation that maps concept descriptions to sets of individuals is quickly identified: the usual interpretation function of the model. Finding an operator that goes in the other direction is not as obvious. We show that the *model-based most specific concept* can be used for this purpose. For a given set of individuals A

the model-based most specific concept is defined to be the most specific concept description that has all elements of A in its interpretation. We show that in $\mathcal{EL}^\bot$ sets of individuals need not have a most specific concept. However, if we extend $\mathcal{EL}^\bot$ to $\mathcal{EL}^\bot_{\text{gfp}}$ by allowing for cyclic concept descriptions interpreted with greatest-fixpoint semantics, then model-based most specific concepts always exist in the resulting logic.

Finding a Finite Base The ontology completion algorithms by Baader et al. use a relatively weak notion of completeness. They consider completeness with respect to a special type of GCIs $C \sqsubseteq D$ where both C and D are conjunctions over previously selected concepts. We adopt a stronger notion of completeness, where we say that a set of GCIs $\mathcal{B}$ is complete for the GCIs holding in a model i if all GCIs $C \sqsubseteq D$ that hold in i follow from $\mathcal{B}$. We make no restriction on C and D other than that they be $\mathcal{EL}^\bot_{\text{gfp}}$-concept descriptions. If the GCIs from $\mathcal{B}$ also hold in the domain then we call $\mathcal{B}$ a *base* for the GCIs holding in i. In general, for each model there are infinitely many GCIs that hold in it. Therefore, it is not obvious whether a base can be finite.

We show that one can always find a finite base $\mathcal{B}$ and we present a construction for it. Model-based most specific concepts are used to compute the right-hand sides of the GCIs in this base. This parallels the Duquenne-Guigues Base in FCA, where the right-hand sides of the implications are obtained by applying the derivation operators to the left-hand sides. Moreover, the left-hand sides occuring in the finite base $\mathcal{B}$ are of a special form: Each of them is a conjunction over a subset of M_i, where M_i is a finite set of concept descriptions that depends on the model i. The fact that $\mathcal{B}$ is finite and complete answers the question for existence of a finite base. Because of the subset construction from which $\mathcal{B}$ is obtained it can, however, be large. To mend this we create the induced context whose attribute set is M_i, and reduce the size of $\mathcal{B}$ by computing the Duquenne-Guigues Base in this induced context. We show that the resulting base is still complete, and even has minimal cardinality among all bases for the GCIs holding in the domain. It is therefore comparable to the Duquenne-Guigues Base. FCA is used twice for this result: The first time it is used indirectly in the form of the model-based most specific concept which is an emulation of the FCA derivation operator in a DL world. The second time it is used directly,

when we reduce the size of the base. In a last step, we show that cyclic descriptions can be removed from $\mathcal{B}$ by unravelling and pruning cyclic descriptions. This yields a finite base for the logic $\mathcal{EL}^{\bot}$ which does not allow for cyclic concept descriptions.

Model Exploration We present *Model Exploration* which is an exploration formalism where counterexamples are stored in a model. The finite base $\mathcal{B}$ can only be computed if the model i is known entirely. It would be overly optimistic to assume that a model which represents the entire domain of the knowledge base is always available. In practice, it is more likely that only fragments of such a model can be obtained. If these fragments are too restricted the GCIs that are computed can only serve as a starting point. The knowledge engineer might have to weaken or even remove some of them. As an example, assume that the domain consists of two families: Kirk, his wife Luann, and their son Milhouse, as well as Jackie, her husband Clancy, and their daughter Selma. The GCI

$$\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top,$$

which says that every father is male and has a child, holds in the domain. If we had used a model consisting only of the first family then the GCI

$$\mathsf{Father} \sqsubseteq \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\mathsf{Male}$$

stating that every father is male and has a male child would also hold. Of course, the second GCI is too specific and the expert will need to modify it. Using Model Exploration she can do this in a fashion that is known from Attribute Exploration: The algorithm successively selects GCIs for which none of the known individuals serve as counterexamples. These axioms are presented to the expert, who can either accept them or refute them by providing a counterexample. In our example, the expert would choose to provide Clancy as the counterexample. As in classical Attribute Exploration, the expert is assumed to have complete knowledge about the domain.

On the technical side, we need to deal with the fact that the GCIs in $\mathcal{B}$ depend on the set M_i which in turn depends on the model i. Since initially only a fragment of i is known the set M_i cannot be computed in advance. Instead, more of its elements become known gradually as the exploration progresses. Therefore, on the FCA level, an algorithm is

needed that allows attributes to be added to a context during runtime. We present such an algorithm and use it as the foundation of Model Exploration.

ABox Exploration When one thinks of an ontology completion formalism, one typically thinks of a formalism that starts with an ontology, and adds axioms until a certain type of completeness is reached. Unfortunately, Model Exploration uses a model, not an ontology as the starting point. A second issue with Model Exploration arises when counterexamples are provided by the expert. Assume that in the above model of two families the expert wants to add Clancy as a counterexample to a GCI that does not hold in the domain, e. g.

$$\mathsf{Father} \sqsubseteq \mathsf{Female}.$$

If she adds only Clancy, but not Selma, then she achieves that while Clancy is a counterexample to the GCI $\mathsf{Father} \sqsubseteq \mathsf{Female}$ he also becomes a counterexample to the GCI

$$\mathsf{Father} \sqsubseteq \exists \mathsf{hasChild}.\top.$$

The problem is that the latter GCI does hold in the domain and one would not want to have a counterexample. Hence, if the expert wants to add Clancy then she also has to add all of Clancy's role successors, which can be a large number. This problem is caused by the closed-world semantics of the model and can be resolved by allowing open-world descriptions of counterexamples. We develop an exploration formalism called *ABox Exploration* where counterexamples are stored in ABoxes, because ABoxes are already part of existing ontologies, and their open-world semantics facilitate the addition of counterexamples. The ontology that serves as a starting point for the exploration should be written in $\mathcal{EL}^{\bot}$. We show that the expressivity of $\mathcal{EL}^{\bot}$ suffices to describe counterexamples.

Since no models are used in ABox Exploration model-based most specific concepts cannot be used to compute the right-hand sides of GCIs, as was the case in Model Exploration. Hence, we need to find a replacement. We show that a suitable replacement are *minimal possible consequences*. For a given concept description C a minimal possible consequence D is a most specific concept description such that adding the

GCI $C \sqsubseteq D$ does not make the ontology inconsistent. Using minimal possible consequences we develop a first version of ABox Exploration.

Unfortunately, unlike model-based most specific concepts, minimal possible consequences are not unique, and they are difficult to compute. To mend this we propose an alternative variant of ABox Exploration that uses an approximation of minimal possible consequences. This approximation has better computational properties, but may result in slightly increased expert interaction.

2 Description Logics

This chapter introduces fundamental notions from Description Logics [BCM+03]. We explain how knowledge is represented in a Description Logic knowledge base. The main focus lies on the lightweight logic $\mathcal{EL}$ and some of its extensions.

In Section 2.1 we introduce the basic syntax and semantics of concept descriptions in the language $\mathcal{EL}$. We also give a short outlook on some of the constructors that are used by more expressive languages. Concept descriptions that are written in a DL language are the building blocks for ontological axioms. An ontology contains axioms which can be grouped into statements about concepts and statements about individuals. According to this distinction ontologies are divided into a terminological part called TBox and an assertional part called ABox. These are introduced in Section 2.2. Classical reasoning services mainly serve to extract implicit knowledge from an ontology (standard reasoning), while newer reasoning services have been designed specifically to assist in the process of ontology engineering (non-standard reasoning). The inference problems that give rise to the most common standard reasoning services are defined in Section 2.3. In addition we present two non-standard reasoning problems that are closely related to this work, namely the most specific concept and the least common subsumer.

In some of the later parts of this thesis the expressivity of $\mathcal{EL}$ will be insufficient. Therefore we introduce extensions of this language in Section 2.4. Among these are $\mathcal{EL}^{\bot}$ and the language $\mathcal{EL}^{\bot}_{\text{gfp}}$ which allows for cyclic concept descriptions.

2.1 Concept Descriptions in $\mathcal{EL}$

As mentioned previously, Description Logics are a family of numerous formal languages for knowledge representation. A prototypical DL knowledge base is a set of axioms that are expressed in a formally defined

DL language. These axioms represent the knowledge that has been specified explicitly by a knowledge engineer. All DL languages use a formal syntax and semantics. Three disjoint, finite sets, namely the set of concept names $\mathcal{N}_C$, the set of role names $\mathcal{N}_R$, and the set of individuals $\mathcal{N}_I$ are used as a starting point for defining DL syntax. Each DL language provides a set of concept constructors that can be used to inductively build concept descriptions from $\mathcal{N}_C$ and $\mathcal{N}_R$.[1] $\mathcal{EL}$ is among the simplest Description Logics, as it provides only three constructors which are the top concept ($\top$), conjunction ($\sqcap$) and existential restrictions ($\exists$).

Definition 2.1 (Syntax of $\mathcal{EL}$ concept descriptions)**.** Let $\mathcal{N}_C$ and $\mathcal{N}_R$ be disjoint sets. Then $\mathcal{EL}$ *concept descriptions* are defined inductively as follows.

- All concept names $A \in \mathcal{N}_C$ and the top concept $\top$ are $\mathcal{EL}$ concept descriptions, and
- if C and D are $\mathcal{EL}$ concept descriptions then $C \sqcap D$ is an $\mathcal{EL}$ concept description, and
- if C is an $\mathcal{EL}$ concept description and $r \in \mathcal{N}_R$ is a role name then $\exists r.C$ is an $\mathcal{EL}$ concept description.

The pair $(\mathcal{N}_C, \mathcal{N}_R)$ is called the *signature* of the concept descriptions. In the following we assume that $\mathcal{N}_C$ and $\mathcal{N}_R$ are always finite.

In general, we stick to the convention that the letters A and B denote concept names, while the letters C, D, E and F denote complex concept descriptions. Roles are always denoted by the lower case letters r, s, and t, while individual names range over the letters a, b, c, etc. In later chapters when we combine Formal Concept Analysis and Description Logics this may interfere with notational conventions from FCA, in which case we may make some exceptions.

The *role depth* $d(C)$ of an $\mathcal{EL}$-concept description C is the maximal number of nestings of existential quantifiers. More formally, it can be

[1] In some of the more expressive logics, individual names from $\mathcal{N}_I$ are also used to build concept descriptions. In the less expressive logics they only occur in ABoxes (cf. Section 2.2).

defined inductively as follows.

$$d(C) = \begin{cases} 0 & \text{if } C = \top \text{ or } C \in \mathcal{N}_P, \\ 1 + d(D) & \text{if } C = \exists r.D, \\ \max\{d(D), d(E)\} & \text{if } C = D \sqcap E. \end{cases}$$

Example 2.1. As an example let the signature be defined as $(\mathcal{N}_C, \mathcal{N}_R)$, where $\mathcal{N}_C = \{\mathsf{Husband}, \mathsf{Wife}, \mathsf{Male}, \mathsf{Female}\}$ and $\mathcal{N}_R = \{\mathsf{marriedTo}\}$. Then

$$\mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\top \tag{2.1}$$

would be a syntactically correct $\mathcal{EL}$-concept description. It has role depth $d(\mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\top) = 1$ since there is only one existential quantifier.

The semantics of DL languages is defined using interpretations.

Definition 2.2 (Interpretations in $\mathcal{EL}$)**.** Let $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$ be a signature. Let i be a pair $i = (\Delta_i, \cdot^i)$ where Δ_i is a non-empty set and $\cdot^i$ is a mapping that maps every concept name $A \in \mathcal{N}_C$ to a subset $A^i \subseteq \Delta_i$, every role name r to a binary relation $r^i \subseteq \Delta_i \times \Delta_i$, and every individual name $a \in \mathcal{N}_I$ to an individual $a^i \in \Delta_i$.

The interpretation function $\cdot^i$ of i can be extended inductively. We denote the extended interpretation function by $\cdot^i$ as well. Let C be a concept description. We define

- If $C = \top$ then $C^i = \Delta_i$, and
- if $C = D \sqcap E$ for some $\mathcal{EL}$-concept descriptions D and E then $C^i = D^i \cap E^i$, and
- if $C = \exists r.D$ for some $\mathcal{EL}$-concept description D and some role name $r \in \mathcal{N}_R$ then $C^i = \{x \in \Delta_i \mid \exists y \in D^i \colon x r^i y\}$.

The pair $i = (\Delta_i, \cdot^i)$ is called an *($\mathcal{EL}$-)interpretation* over the signature $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$. The set Δ_i is called the *domain* of i. We call the elements of Δ_i *individuals*.[2] For a concept description C the set C^i is called the *extension* of C.

[2] This differs slightly from the usual terminology, where the term *individual* is only used for individuals in an ABox, and $x \in \Delta_i$ is simply referred to as an *element of the domain* of i.

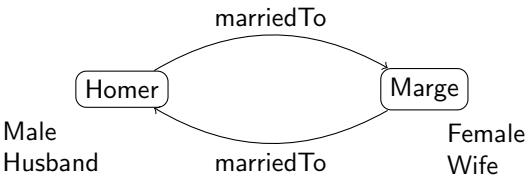

Figure 2.1: An Interpretation Describing a Married Couple

Example 2.2. Consider the pair $i_S = (\Delta_{i_S}, \cdot^{i_S})$ where

$$\begin{aligned}
\Delta_{i_S} &= \{\mathsf{Homer}, \mathsf{Marge}\},\\
\mathsf{Male}^{i_S} = \mathsf{Husband}^{i_S} &= \{\mathsf{Homer}\},\\
\mathsf{Female}^{i_S} = \mathsf{Wife}^{i_S} &= \{\mathsf{Marge}\},\\
\mathsf{marriedTo}^{i_S} &= \{(\mathsf{Homer}, \mathsf{Marge}), (\mathsf{Marge}, \mathsf{Homer})\},\\
\mathsf{Homer}^{i_S} &= \mathsf{Homer},\\
\mathsf{Marge}^{i_S} &= \mathsf{Marge}.
\end{aligned}$$

Then i_S is an interpretation over the signature $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$ where $\mathcal{N}_C$ and $\mathcal{N}_R$ are as in Example 2.1 and $\mathcal{N}_I = \{\mathsf{Homer}, \mathsf{Marge}\}$. It is depicted in Figure 2.1. The interpretation function $\cdot^{i_S}$ maps the concept description (2.1) to

$$\begin{aligned}
(\mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\top)^{i_S} &= \mathsf{Male}^{i_S} \cap (\exists \mathsf{marriedTo}.\top)^{i_S} =\\
&= \{\mathsf{Homer}\} \cap \{\mathsf{Homer}, \mathsf{Wife}\} = \{\mathsf{Homer}\}. \quad (2.2)
\end{aligned}$$

Other languages provide more than the three constructors from $\mathcal{EL}$. A logic of particular relevance is the Description Logic $\mathcal{ALC}$ which provides for $\top$, $\bot$, conjunction, disjunction, existential restrictions, value restrictions and negation. Table 2.1 lists the constructors from $\mathcal{ALC}$ along with their syntax and semantics. $\mathcal{ALC}$ is the smallest Boolean closed DL. It is important from a theoretical point of view as it has helped to establish the connection between DL and modal logics [BdRV01]. In [Sch91] it is shown that $\mathcal{ALC}$ can be viewed as a notational variant of the multimodal

Table 2.1: Syntax and Semantics of Various Concept Constructors

Name	Syntax	Semantics
top concept	$\top$	Δ_i
bottom concept	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^i \cap D^i$
disjunction	$C \sqcup D$	$C^i \cup D^i$
negation	$\neg C$	$\Delta_i \setminus C^i$
existential restrictions	$\exists r.C$	$\{x \in \Delta_i \mid \exists y \in C^i \colon (x, y) \in r^i\}$
value restrictions	$\forall r.C$	$\{x \in \Delta_i \mid \forall y \in \Delta_i \colon (x, y) \in r^i \Rightarrow y \in C^i\}$
nominals	$\{a\}$	$\{a^i\}$

logic $\mathbf{K}_{(\mathbf{m})}$. When we speak of *lightweight Description Logics* we usually mean Description Logics that do not provide all the constructors from $\mathcal{ALC}$. Conversely, we say *expressive Description Logics* when we talk about logics that provide more constructors than $\mathcal{ALC}$.

2.2 Ontologies

In the previous section we have shown how concept descriptions are generated. In this section we show how concept descriptions can be used to express facts about the domain of interest. For example in an ontology about human genealogy we might want to express that the concept description from (2.1) describes the concept Husband. In DL syntax this can be formulated as

$$\mathsf{Husband} \equiv \mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\top. \tag{2.3}$$

This expression states something about the terminology of the domain, it explains the concept Husband using other concept names and a role name. In a typical DL ontology this kind of statement is stored in a TBox, which is short for *terminological box*. We distinguish three types of TBoxes: acyclic TBoxes, cyclic TBoxes and general TBoxes.

Definition 2.3 (Acyclic TBox)**.** Let A be a concept name and C a concept description. Then $A \equiv C$ is called a *concept definition*. A finite set $\mathcal{T}$ of concept definitions is called an *acyclic TBox* if

- for every concept name $A \in \mathcal{N}_C$ there is at most one concept description C such that $A \equiv C \in \mathcal{T}$, and
- there is no sequence of concept descriptions $A_1 \equiv C_1, \ldots, A_n \equiv C_n$ in $\mathcal{T}$ where A_i occurs in C_{i-1} for all $i \in \{2, \ldots, n\}$ and A_1 occurs in C_n.

In other words in an acyclic TBox every concept name $A \in \mathcal{N}_C$ can be defined at most once, and its definition is not allowed to use the name A explicitly or implicitly. Another name for acyclic TBox is *unfoldable TBox*.

Definition 2.4 (Cyclic TBox)**.** A *cyclic TBox* is a finite set of concept definitions $\mathcal{T}$ where for every concept name $A \in \mathcal{N}_C$ there is at most one concept description C such that $A \equiv C \in \mathcal{T}$.

Definition 2.5 (General TBox)**.** Let C and D be concept descriptions. The statement $C \sqsubseteq D$ is called a *general concept inclusion (GCI)*. A *general TBox* is a finite set of GCIs.

Using TBoxes we can express knowledge about concepts, such as the definition of the concept Husband (2.3). DL knowledge bases usually also contain knowledge about individuals which is stored in an ABox. Using ABoxes, one can express that an individual belongs to a concept:

$$\mathsf{Husband}(\mathsf{Homer})$$

expresses that the individual Homer belongs to the concept Husband. One can also express that two individuals are in a relation:

$$\mathsf{marriedTo}(\mathsf{Homer}, \mathsf{Marge})$$

expresses that Homer is in relation marriedTo with Marge.

Definition 2.6 (ABox)**.** Let $A \in \mathcal{N}_C$ be a concept name, $r \in \mathcal{N}_R$ a role name and $a, b \in \mathcal{N}_I$ individual names. A *concept assertion* is a statement of the form $A(a)$. Statements of the form $r(a, b)$ are called *role assertions*. A set $\mathcal{A}$ of concept assertions and role assertions is called an *ABox*.

Table 2.2: Semantics of Ontological Statements

Name	Syntax	Semantics
concept definition	$A \equiv C$	$A^i = C^i$
GCI	$C \sqsubseteq D$	$C^i \subseteq D^i$
role assertion	$r(a, b)$	$(a^i, b^i) \in r^i$
concept assertion	$A(a)$	$a^i \in A^i$
role inclusion	$r_1 \circ \cdots \circ r_k \sqsubseteq r$	$r_1^i \circ \cdots \circ r_k^i \subseteq r^i$

DL knowledge bases typically comprise an ABox and a TBox of one of the above types. We call a pair $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ an *ontology*. Some DLs allow for more advanced statements, such as restrictions on roles, in addition to concept definitions, concept assertions and role assertions. Prominent examples of DLs with role restrictions are $\mathcal{EL}^{++}$ and expressive DLs like $\mathcal{SHIQ}$ [BBL05a, HST99]. In the following we present the semantics for the basic types of ABoxes and TBoxes as presented above. A summary of these and a selection of some of the more advanced ontology statements can be seen in Table 2.2.

Definition 2.7 (Semantics of TBoxes)**.** Let $A \equiv C$ be a concept description and $i = (\Delta_i, \cdot^i)$ an interpretation. We say that $A \equiv C$ *holds in* i if i satisfies $A^i = C^i$. Likewise, if $C \sqsubseteq D$ is a GCI, we say that $C \sqsubseteq D$ *holds in* i if i satisfies $C^i \subseteq D^i$.

Let $\mathcal{T}_1$ be an acyclic or cyclic TBox, and let $i = (\Delta_i, \cdot^i)$ be an interpretation. We call i a *model of* $\mathcal{T}_1$ if every concept definition $A \equiv C \in \mathcal{T}_1$ holds in i. Let $\mathcal{T}_2$ be a general TBox. We call i a *model of* $\mathcal{T}_2$ if every GCI $C \sqsubseteq D \in \mathcal{T}$ holds in i.

While there is only one widely accepted semantics for acyclic and general TBoxes there are several possibilities for cyclic TBoxes. In [Neb91] Nebel describes three possible types of semantics for cyclic TBoxes: descriptive semantics, greatest-fixpoint semantics and least-fixpoint semantics. The semantics presented in Definition 2.7 are called *descriptive semantics*. In the Section 2.4.1 we introduce *greatest-fixpoint semantics*. The third type of semantics, *least-fixpoint semantics*, is not relevant for this work and therefore omitted.

Let $i = (\Delta_i, \cdot^i)$ be an interpretation. To define the semantics of ABoxes we extend $\cdot^i$ to map every individual name $a \in \mathcal{N}_I$ to an element $a^i \in \Delta_i$.

Definition 2.8 (Semantics of ABoxes)**.** Let $(\mathcal{N}_C, \mathcal{N}_R)$ be a signature, $\mathcal{A}$ an ABox and i be an interpretation. We call i a *model* of $\mathcal{A}$ if

- for every concept assertion $A(a) \in \mathcal{A}$ it holds that $a^i \in A^i$, and
- for every role assertion $r(a, b) \in \mathcal{A}$ it holds that $(a^i, b^i) \in r^i$.

The *models of an ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ are those interpretations that are models of both $\mathcal{T}$ and $\mathcal{A}$.

Example 2.3. Let $\mathcal{O}_F = (\mathcal{T}_F, \mathcal{A}_F)$ be an ontology that consists of the TBox

$$\mathcal{T}_F = \{\mathsf{Husband} \equiv \mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\top, \\ \mathsf{Wife} \equiv \mathsf{Female} \sqcap \exists \mathsf{marriedTo}.\top\}$$

and the ABox $\mathcal{A}_F = \{\mathsf{Husband}(\mathsf{Homer})\}$. Then the interpretation i_S from Figure 2.1 is a model for $\mathcal{O}_F$ because both concept definitions from $\mathcal{T}_F$ as well as the concept assertion from $\mathcal{A}_F$ hold in i_S.

Notice that i_S would still be a model if we removed the concept assertion from $\mathcal{A}_F$. If $\mathcal{A}$ does not contain the statement $\mathsf{Husband}(a)$ then it is not automatically assumed that the contrary holds. Due to this behaviour the semantics of DL ontologies are often called *open-world* semantics.

2.3 Reasoning in DL

The strength of logic based formalisms for knowledge representation such as DL is the possibility to make *implicit* knowledge explicit through reasoning services. Implicit knowledge is knowledge that is not present in the form of an axiom (a TBox or ABox statement in our setting), but can be inferred logically from the axioms in the knowledge base. Originally, DL research focussed on a class of reasoning services that are now called *standard reasoning services*. The most important standard reasoning services are

- concept subsumption,
- instance checking,
- concept satisfiability, and
- consistency checking.

In this section we give formal definitions of these reasoning services for a generic DL.

Definition 2.9 (Subsumption)**.** Let C and D be concept descriptions. We say that D *subsumes* C and write $C \sqsubseteq D$ if for all interpretations i the GCI $C \sqsubseteq D$ holds in i, i. e. if all interpretations i satisfy $C^i \subseteq D^i$.

Let $\mathcal{O}$ be an ontology, consisting of an ABox $\mathcal{A}$ or a TBox $\mathcal{T}$ or both. We say that D *subsumes* C *with respect to* $\mathcal{O}$ and write $C \sqsubseteq_{\mathcal{O}} D$ if $C \sqsubseteq D$ holds in all models i of $\mathcal{O}$. If D subsumes C with respect to $\mathcal{O}$ then we also say that $C \sqsubseteq D$ *follows from* $\mathcal{O}$.

The identification of all subsumptions between the concept names that occur in an ontology $\mathcal{O}$ is called the *classification* of $\mathcal{O}$. The relation $\sqsubseteq$ forms a preorder on the set of all concept descriptions. We say that two concept descriptions C and D are *equivalent* if both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold. Equivalence of C and D is denoted by $C \equiv D$.

Definition 2.10 (Instance Checking)**.** Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology, let C be a concept description and $a \in \mathcal{N}_I$ an individual. Then a *is an instance of* C if $a^i \in C^i$ holds for all models i of $\mathcal{O}$. This is denoted by $\mathcal{O} \models C(a)$.

Definition 2.11 (Concept Satisfiability)**.** Let $\mathcal{O}$ be an ontology and C a concept description. C is *satisfiable* if there is a model i of $\mathcal{O}$ such that C^i is not empty. Otherwise C is *unsatisfiable*.

Definition 2.12 (Consistency Checking)**.** Let $\mathcal{O}$ be an ontology. $\mathcal{O}$ is *consistent* if $\mathcal{O}$ has a model.

Consistency Checking and Satisfiability Checking can be used to detect the most serious errors in the knowledge base, namely those errors that render a concept description or a whole ontology unusable. These

errors are only the tip of the iceberg, since it is still possible for a satisfiable concept description to misrepresent the knowledge engineer's intention.

The two reasoning services *consistency checking* and *concept satisfiability* are not applicable to a Description Logic without a form of negation, i. e. in particular to $\mathcal{EL}$. For example in $\mathcal{EL}$ all ontologies have a model $i = (\Delta_i, \cdot^i)$ with $\Delta_i = \mathcal{N}_I$, $a^i = a$ for all $a \in \mathcal{N}_I$, $A^i = \Delta_i$ for all $A \in \mathcal{N}_C$ and $r^i = \Delta_i \times \Delta_i$ for all $r \in \mathcal{N}_R$. The same model satisfies all $\mathcal{EL}$-concept descriptions over the given signature.

The typical usage scenario for standard reasoning services is when a user wants to derive implicit knowledge from an already existing ontology. There are other reasoning services, the *non-standard reasoning services* that have been designed with the intention to assist in the process of ontology design and maintenance. This distinction is not strict, since certain standard reasoning services can be used to detect errors in an existing ontology and thereby assist in the maintenance process.

We present two non-standard reasoning services that are relevant to this work. These are

- least common subsumers, and
- most specific concepts.

Other non-standard reasoning services include *matching*, *pinpointing* or *modularization* [BK06, SC03, Peñ09, GHKS07].

The idea behind the most specific concept is to capture the complete knowledge about an individual in an ontology in a single concept description. Least common subsumers are used to extract commonalities from two concept descriptions.

Definition 2.13 (Most Specific Concept)**.** Let $\mathcal{O}$ be an ontology and $a \in \mathcal{N}_I$ an individual name. Let E be a concept description satisfying $\mathcal{O} \models E(a)$ such that for all concept descriptions F it holds that $\mathcal{O} \models F(a)$ implies $E \sqsubseteq F$. Then E is called the *most specific concept* of a with respect to $\mathcal{O}$.

Definition 2.14 (Least Common Subsumer)**.** Let $\{C_1, C_2, \ldots, C_n\}$ be a finite set of concept descriptions. Let E be a concept description such that

- $C_k \sqsubseteq E$ holds for all $k \in \{1, \ldots, n\}$, and
- for all concept descriptions F it holds that $C_k \sqsubseteq F$ for all $k \in \{1, \ldots, n\}$ implies $E \sqsubseteq F$.

Then E is called the *least common subsumer* of $\{C_1, C_2, \ldots, C_n\}$.

Least common subsumers and most specific concepts need not exist for all logics. If they exist both are unique up to equivalence. A typical scenario for the application of most specific concepts and least common subsumers is the bottom-up approach to knowledge engineering [BT01]. Imagine that an ontology engineer is working on an incomplete ontology. She is trying to come up with a description for a difficult concept. From the ABox she can select an existing individual, that belongs to the concept that she wants to describe, and compute its most specific concept. Most likely this description is more specific than what she had in mind. She can then choose another individual, compute its most specific concept and extract the commonalities from the two most specific concepts using least common subsumers.

2.4 $\mathcal{EL}$ and its Offspring

We concentrate on the Description Logic $\mathcal{EL}$ and some of its extensions. The most important one among the reasons for this choice is that the standard reasoning problems are tractable. This is not the case for expressive Description Logics and even some lightweight Description Logics. A first algorithm for the classification of $\mathcal{EL}$-ontologies with general TBoxes and role inclusion axioms has been proposed in [Bra04]. This algorithm has been extended to the more powerful DL $\mathcal{EL}^{++}$ in [BBL05a], and it has been generalized even further in [BBL08]. A refined version of it was the subject of [Sun09]. $\mathcal{EL}^{++}$ allows for the bottom concept, nominals, concrete domains[3] and role inclusions in addition to the concept constructors from $\mathcal{EL}$. Because $\mathcal{EL}^{++}$ is tractable every fragment of $\mathcal{EL}^{++}$ is also tractable. In particular, if we extend $\mathcal{EL}$ by adding the bottom concept $\bot$ we obtain the simple tractable DL $\mathcal{EL}^{\bot}$.

[3] Concrete Domains are not formally introduced because they are not relevant for this work. We refer the interested reader to [Lut03] for more information.

Definition 2.15 ($\mathcal{EL}^\bot$)**.** $\mathcal{EL}^\bot$-concept description are all concept descriptions that can be generated inductively using the constructors $\top$, $\bot$, existential restrictions, and conjunction, where syntax and semantics are as described in Table 2.1.

A disadvantage of $\mathcal{EL}^\bot$ is that it cannot express cyclic dependencies. Some of the methods that we present later are therefore not applicable to $\mathcal{EL}^\bot$ directly. Instead we need to use the more expressive logic $\mathcal{EL}^\bot_{\text{gfp}}$.

2.4.1 Extending $\mathcal{EL}$ by Terminological Cycles

In Section 2.2 we have introduced the usual semantics of DL knowledge bases, that is also called *descriptive semantics*. As mentioned previously, there are other ways to define the semantics of cyclic TBoxes, the so-called *greatest-fixpoint semantics*. To define fixpoint semantics we need to distinguish between the defined and the primitive concept names of a TBox.

Definition 2.16 (Primitive Concept Names and Primitive Interpretations)**.** Let $\mathcal{T}$ be a cyclic $\mathcal{EL}$ TBox and $(\mathcal{N}_C, \mathcal{N}_R)$ its signature. A concept name $A \in \mathcal{N}_C$ is called a *primitive concept name* if there is no concept definition $A \equiv C$ in $\mathcal{T}$ for any concept description C. A is called a *defined concept name* otherwise. We denote the set of all primitive concept names of $\mathcal{T}$ by $\mathcal{N}_P(\mathcal{T})$ and the set of all defined concept names of $\mathcal{T}$ by $\mathcal{N}_D(\mathcal{T})$.

A *primitive interpretation of* $\mathcal{T}$ is a pair $i = (\Delta_i, \cdot^i)$ consisting of a non-empty set Δ_i and and interpretation function $\cdot^i$. The function $\cdot^i$ maps every primitive concept name $A \in \mathcal{N}_P(\mathcal{T})$ to a set $A^i \subseteq \Delta_i$, and every role name $r \in \mathcal{N}_R$ to a binary relation $r \subseteq \Delta_i \times \Delta_i$ (and every individual name $a \in \mathcal{N}_I$ to an individual $a^i \in \Delta_i$ if individual names are present). The only difference between an interpretation and a primitive interpretation is that the latter does not interpret the defined concepts. We say that an interpretation $j = (\Delta_j, \cdot^j)$ *extends* a primitive interpretation $i = (\Delta_i, \cdot^i)$ if $\Delta_i = \Delta_j$ and $A^i = A^j$ for all primitive concept names $A \in \mathcal{N}_P(\mathcal{T})$ and $r^i = r^j$ for all role names $r \in \mathcal{N}_R$ (and $a^i = a^j$ for all $a \in \mathcal{N}_I$ if individual names are present).

Example 2.4. Let us assume that the knowledge engineer who is working on the TBox $\mathcal{T}_F$ from Example 2.3 has decided that she wants to

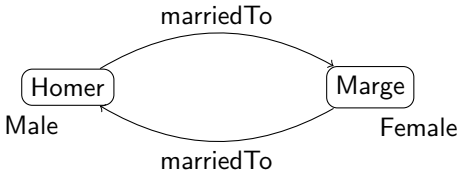

Figure 2.2: A Primitive Interpretation for $\mathcal{T}'_F$ from Example 2.4

ensure that a husband is always married to a wife and a wife is always married to a husband. She might change the TBox to read

$$\begin{aligned}\mathcal{T}'_F = \{&\mathsf{Husband} \equiv \mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\mathsf{Wife},\\ &\mathsf{Wife} \equiv \mathsf{Female} \sqcap \exists \mathsf{marriedTo}.\mathsf{Husband}\}.\end{aligned}$$

These changes cause the TBox to become cyclic. In this TBox $\mathcal{T}'_F$ Husband and Wife are defined concept names, while Male and Female are primitive concept names. Therefore $i = (\{\mathsf{Homer}, \mathsf{Marge}\}, \cdot^i)$ with

$$\begin{aligned}\mathsf{Male}^i &= \{\mathsf{Homer}\}\\ \mathsf{Female}^i &= \{\mathsf{Marge}\}\\ \mathsf{marriedTo}^i &= \{(\mathsf{Homer}, \mathsf{Marge}), (\mathsf{Marge}, \mathsf{Homer})\}\end{aligned}$$

is a primitive interpretation of $\mathcal{T}$. A graphical representation of i is shown in Figure 2.2. Using the descriptive semantics from Definition 2.7 we can find two models j_1 and j_2 of $\mathcal{T}_F$ that are both extensions of i. These two models are defined by $\mathsf{Husband}^{j_1} = \emptyset$, $\mathsf{Wife}^{j_1} = \emptyset$, and $\mathsf{Husband}^{j_2} = \{\mathsf{Homer}\}$, $\mathsf{Wife}^{j_2} = \{\mathsf{Marge}\}$. Most likely only j_2, which is identical to i_S from Figure 2.1 captures the knowledge engineer's intuition.

For a given primitive interpretation greatest fixpoint semantics allow only models that map each defined concept name to the largest possible set. To define this formally and to prove existence of these so-called greatest-fixpoint models (gfp-models) one uses Tarski's theorem [DP02, Bir93].

Definition 2.17 (Complete Lattices and Order Preserving Mappings)**.** A ordered set $(L, \leq)$ is called a *complete lattice* if every set $S \subseteq L$ has a least upper bound and a greatest lower bound. A mapping $f\colon L \to L$ is called *order-preserving* if $x \leq y$ implies $f(x) \leq f(y)$.

Theorem 2.1 (Tarski)**.** *Let L be a complete lattice and $f\colon L \to L$ an order-preserving mapping. Then f has a least fixpoint and a greatest fixpoint.*

Let $\mathcal{T}$ be a cyclic TBox and i a primitive interpretation. By Ext_i we denote the set of all interpretations of $\mathcal{T}$ that extend i. $\mathcal{T}$ gives rise to a function $\mathcal{T}_i\colon \mathrm{Ext}_i \to \mathrm{Ext}_i$ that is defined as follows. Let $j \in \mathrm{Ext}_i$ be an interpretation. Then we define $\mathcal{T}_i(j)$ to be the interpretation $\mathcal{T}_i(j) = (\Delta_i, \cdot^{\mathcal{T}_i(j)})$ where

$$A^{\mathcal{T}_i(j)} = C^j \qquad \text{for all defined concept names } A \in \mathcal{N}_D(\mathcal{T}), \text{ where } A \equiv C \in \mathcal{T} \tag{2.4}$$

$$A^{\mathcal{T}_i(j)} = A^i \qquad \text{for all primitive concept names } A \in \mathcal{N}_P(\mathcal{T}) \tag{2.5}$$

$$r^{\mathcal{T}_i(j)} = r^i \qquad \text{for all role names } r \in \mathcal{N}_R. \tag{2.6}$$

Notice that $j \in \mathrm{Ext}_i$ is a model of $\mathcal{T}$ in the sense of Definition 2.7 if and only if $A^j \equiv C^j$ holds for all defined concept names $A \in \mathcal{N}_D(\mathcal{T})$. Using (2.4) we can rewrite this to read $A^j \equiv A^{\mathcal{T}_i(j)}$ for all defined concept names. It follows that j is a model of $\mathcal{T}$ in the sense of Definition 2.7 if and only if $j = \mathcal{T}_i(j)$ holds, i. e. the models of $\mathcal{T}$ are the fixpoints of $\mathcal{T}_i$.

In order to be able to speak of a *greatest* fixpoint we define an order on Ext_i. Let j_1 and j_2 be two interpretations from Ext_i. We write $j_1 \leq j_2$ if all concept names $A \in \mathcal{N}_D(\mathcal{T})$ satisfy $A^{j_1} \subseteq A^{j_2}$. It can be shown that $\mathcal{T}_i$ is order-preserving with respect to $\leq$. Hence, Tarki's Theorem shows that $\mathcal{T}_i$ has a greatest fixpoint for every primitive interpretation i. We call the greatest fixpoint of $\mathcal{T}_i$ a *greatest fixpoint model (gfp-model)* of $\mathcal{T}$.

In Example 2.4 the TBox $\mathcal{T}'_F$ has exactly two fixpoints, namely the models j_1 and j_2. The second interpretation j_2 that interprets Husband as {Homer} and and Wife as {Marge} is a gfp-model.

We now have the necessary tools to define the syntax and semantics of $\mathcal{EL}_{\mathrm{gfp}}$.

Definition 2.18 (Syntax of $\mathcal{EL}_{\text{gfp}}$)**.** We say that a cyclic $\mathcal{EL}$-TBox $\mathcal{T}$ is *normalized* if every concept definition in $\mathcal{T}$ is of the form

$$B \equiv P_1 \sqcap \cdots \sqcap P_k \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_l.B_l,$$

where B, B_1, ..., B_l are defined concept names, r_1, ..., r_l are role names and P_1, ..., P_k are primitive concept names.

An $\mathcal{EL}_{\text{gfp}}$-*concept description* is a pair $C = (A_C, \mathcal{T}_C)$ where $\mathcal{T}_C$ is a normalized $\mathcal{EL}$-TBox and A_C is a defined concept name from $\mathcal{T}_C$. A_C is called the *root concept* of C. The signature of C is the pair $(\mathcal{N}_P, \mathcal{N}_R)$, where $\mathcal{N}_P$ is the set of primitive concept names that are used in $\mathcal{T}_C$ and $\mathcal{N}_R$ is the set of role names.

Notice that we do not include the defined concept names in the signature of an $\mathcal{EL}_{\text{gfp}}$-concept description. This is because we will view the pair $(A_C, \mathcal{T}_C)$ as a black box, where we are not interested in the interpretation of each defined concept, but only of the concept description as a whole.

Definition 2.19 (Semantics of $\mathcal{EL}_{\text{gfp}}$)**.** Let $i = (\Delta_i, \cdot^i)$ be a primitive interpretation for a set of primitive concept names $\mathcal{N}_P$ and a set of role names $\mathcal{N}_R$. We extend i to interpret every $\mathcal{EL}_{\text{gfp}}$-concept description $C = (A_C, \mathcal{T}_C)$ over the signature $(\mathcal{N}_P, \mathcal{N}_R)$ as

$$C^i = A_C^j,$$

where j is the gfp-model of $\mathcal{T}_C$ that extends i.

Since the TBoxes within a pair $C = (A, \mathcal{T})$ can be cyclic we can use $\mathcal{EL}_{\text{gfp}}$-concept descriptions to express cyclic dependencies. Unfortunately, the price that we pay is that concept descriptions may become hard to read. In a strict sense even the symbol $\top$ and concept names are not $\mathcal{EL}_{\text{gfp}}$-concept descriptions. For matters of convenience we will abbreviate by $\top$ the $\mathcal{EL}_{\text{gfp}}$-concept description $(A_\top, \{A_\top \equiv \top\})$. Similarly, if $B \in \mathcal{N}_C$ is a concept name we denote by B the concept description $(A_B, \{A_B \equiv B\})$. When $C = (A_C, \mathcal{T}_C)$ and $D = (A_D, \mathcal{T}_D)$ are $\mathcal{EL}_{\text{gfp}}$-concept descriptions it will sometimes be convenient to refer to the concepts $\exists r.C$ or $C \sqcap D$. This is a slight abuse of notation, since C and D are not $\mathcal{EL}$-concept descriptions, but pairs of an ABox and a TBox. However, we can introduce this notation by defining $\exists r.C = (A_{\exists r.C}, \mathcal{T}_{\exists r.C})$,

where $A_{\exists r.C}$ is a new defined concept name and

$$\mathcal{T}_{\exists r.C} = \mathcal{T}_C \cup \{A_{\exists r.C} \equiv \exists r.A_C\}.$$

Similarly we define $C \sqcap D$ to be the pair $C \sqcap D = (A_{C \sqcap D}, \mathcal{T}_{C \sqcap D})$, where $A_{C \sqcap D}$ is a new defined concept name and

$$\mathcal{T}_{C \sqcap D} = \mathcal{T}_C \cup \mathcal{T}_D \cup \{A_{C \sqcap D} \equiv E \sqcap F\},$$

where $A_C \equiv E$ and $A_D \equiv F$ are the concept definitions of A_C and A_D in $\mathcal{T}_C$ and $\mathcal{T}_D$, respectively, and where we assume $\mathcal{N}_D(\mathcal{T}_C)$ and $\mathcal{N}_D(\mathcal{T}_D)$ to be disjoint. The above allows us to inductively translate every $\mathcal{EL}$-concept description into an $\mathcal{EL}_{\text{gfp}}$-concept description. This translation preserves the semantics. Therefore $\mathcal{EL}$ can be viewed as a fragment of $\mathcal{EL}_{\text{gfp}}$.

A simple extension of $\mathcal{EL}_{\text{gfp}}$ is $\mathcal{EL}^{\bot}_{\text{gfp}}$, where we allow for the bottom concept $\bot$ in addition to $\mathcal{EL}_{\text{gfp}}$-concept descriptions.

Definition 2.20 (Syntax and Semantics of $\mathcal{EL}^{\bot}_{\text{gfp}}$)**.** We call C an $\mathcal{EL}^{\bot}_{\text{gfp}}$-*concept description* if either

- $C = \bot$, or
- C is an $\mathcal{EL}_{\text{gfp}}$-concept description.

Let $i = (\Delta_i, \cdot^i)$ be an interpretation. The semantics are defined in the natural way: If $C = \bot$ then $C^i = \emptyset$. Otherwise C is interpreted like a normal $\mathcal{EL}_{\text{gfp}}$-concept description according to Definition 2.19.

2.4.2 Reasoning in $\mathcal{EL}_{\text{gfp}}$

Tarski's Theorem does not provide a practical way to reason in logics that use greatest fixpoint semantics. There is an alternative characterization of $\mathcal{EL}_{\text{gfp}}$ semantics developed by Baader [Baa03b]. It is based on simulations between directed graphs with labelled vertices and edges. Baader calls these graphs $\mathcal{EL}$*-description graphs*.

Definition 2.21 ($\mathcal{EL}$-description graph)**.** An $\mathcal{EL}$*-description graph* is a triple (V, E, L), where

- V is set of vertices, and

- $E \subseteq V \times \mathcal{N}_R \times V$ is a set of edges labelled with role names, and
- $L\colon V \to 2^{\mathcal{N}_C}$ is a function that labels every vertex with a set of concept names.

We have already seen a representation of an $\mathcal{EL}$-interpretation as an $\mathcal{EL}$-description graph, namely the graph from Figure 2.1 that represents the interpretation i_S from Example 2.3. In fact, every $\mathcal{EL}$-interpretation has a corresponding $\mathcal{EL}$-description graph.

Definition 2.22 ($\mathcal{EL}$-Description Graph of an $\mathcal{EL}$-Interpretation)**.** Let $i = (\Delta_i, \cdot^i)$ be an interpretation for the signature $(\mathcal{N}_C, \mathcal{N}_R)$. Let $x \in \Delta_i$ be an individual. By $\text{names}_i(x)$ we denote the set

$$\text{names}_i(x) = \{A \in \mathcal{N}_C \mid x \in A^i\},$$

the set of all concept names that have x in their extension. Let $r \in \mathcal{N}_R$ be a role name. By $\text{succ}_i^r(x)$ denote the set

$$\text{succ}_i^r(x) = \{y \in \Delta_i \mid (x, y) \in r^i\}.$$

The *$\mathcal{EL}$-description graph of i* is the graph $G_i = (\Delta_i, E_i, \text{names}_i)$, where E_i is defined as

$$E_i = \{(x, r, y) \mid r \in \mathcal{N}_R, y \in \text{succ}_i^r(x)\}$$

Conversely, every $\mathcal{EL}$-description graph $G = (V, E, L)$ gives rise to an interpretation $i_G = (V, \cdot^{i_G})$, where $A^{i_G} = \{x \in V \mid A \in L(x)\}$ and $r^{i_G} = \{(x, y) \in V \times V \mid (x, r, y) \in E\}$.

Cyclic $\mathcal{EL}$-TBoxes also have a structure that can be transformed into an $\mathcal{EL}$-description graph.

Definition 2.23 ($\mathcal{EL}$-Description Graph of a TBox)**.** Let $\mathcal{T}$ be a cyclic TBox that uses the set of role names $\mathcal{N}_R$, the set of primitive concept names $\mathcal{N}_P(\mathcal{T})$ and the set of defined concept names $\mathcal{N}_D(\mathcal{T})$. By definition for every defined concept name $B \in \mathcal{N}_D(\mathcal{T})$ the TBox $\mathcal{T}$ contains a concept definition

$$B \equiv P_1 \sqcap \cdots \sqcap P_k \sqcap \exists r_1.B_1 \sqcap \cdots \sqcap \exists r_l.B_l,$$

where B, B_1, ..., B_l are defined concept names, r_1, ..., r_l are role names and P_1, ..., P_k are primitive concept names. By $\text{names}_{\mathcal{T}}(B)$ we denote the set $\text{names}_{\mathcal{T}}(B) = \{P_1, \ldots, P_k\}$ of all primitive concept names that occur in the definition of B. By $\text{succ}^r_{\mathcal{T}}(B)$ we denote the set

$$\text{succ}^r_{\mathcal{T}}(B) = \{A \in \mathcal{N}_D(\mathcal{T}) \mid \exists m \in \{1, \ldots, l\} \colon r_m = r \text{ and } B_m = A\}.$$

We call the graph $G_{\mathcal{T}} = (\mathcal{N}_D(\mathcal{T}), E_{\mathcal{T}}, \text{names}_{\mathcal{T}})$ where $E_{\mathcal{T}}$ is defined as

$$E_{\mathcal{T}} = \{(A, r, B) \mid r \in \mathcal{N}_R, B \in \text{succ}^r_{\mathcal{T}}(A)\}$$

the *$\mathcal{EL}$-description graph of $\mathcal{T}$*. Conversely, every finite $\mathcal{EL}$-description graph $G = (V, E, L)$ gives rise to a TBox $\mathcal{T}_G$, whose defined concept names are $\mathcal{N}_D(\mathcal{T}_G) = V$. For every $A \in V$ the TBox $\mathcal{T}_G$ contains the concept definition $A \equiv \bigsqcap L(A) \sqcap \bigsqcap\{\exists r.B \mid (A, r, B) \in E\}$, where we define $\bigsqcap U$ to be the conjunction over all elements of U for a set U of $\mathcal{EL}$-concept descriptions, and where we define the empty conjunction to be $\top$.

We have mentioned that $\mathcal{EL}$ corresponds to a fragment of $\mathcal{EL}_{\text{gfp}}$. This fragment contains exactly those $\mathcal{EL}_{\text{gfp}}$-concept descriptions whose $\mathcal{EL}$-description graph is a tree. We call those $\mathcal{EL}_{\text{gfp}}$-concept description whose $\mathcal{EL}$-description graph is a tree *acyclic $\mathcal{EL}_{\text{gfp}}$-concept descriptions*. Likewise, we say that an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description C is *acyclic* if either $C = \bot$ or C is an acyclic $\mathcal{EL}_{\text{gfp}}$-concept description.

Baader characterizes instance relations in $\mathcal{EL}_{\text{gfp}}$ using simulations between the description graphs of TBoxes and interpretations.

Definition 2.24 (Simulation)**.** Let $G = (V, E, L)$ and $H = (W, F, M)$ be two $\mathcal{EL}$-description graphs over the same signature $(\mathcal{N}_C, \mathcal{N}_R)$. A binary relation $Z \subseteq V \times W$ is called a *simulation from G to H* if

(S1) every pair $(v, w) \in Z$ satisfies $L(v) \subseteq M(w)$, and

(S2) if $(v, w) \in Z$ and $(v, r, v') \in E$ then there exists some $w' \in W$ such that $(v', w') \in Z$ and $(w, r, w') \in F$.

Lemma 2.2 (Characterizing Instance in $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$)**.** *Let $C = (A, \mathcal{T})$ be an $\mathcal{EL}_{\text{gfp}}$-concept description, let $i = (\Delta_i, \cdot^i)$ be an interpretation and $x \in \Delta_i$ an individual. Then the following two statements are equivalent*

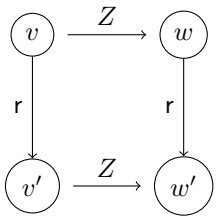

Figure 2.3: An Illustration of Property (S2)

- $x \in C^i$
- *there is a simulation Z from the description graph of $\mathcal{T}$ to the description graph of i such that $(A, x) \in Z$.*

Obviously, the same holds if C is an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description with $C \neq \bot$. If $C = \bot$ then $x \in C^i$ never holds.

For any two $\mathcal{EL}$-description graphs G and H there is a maximal simulation $Z_{\max}$ from G to H. By maximal we mean that if Z is a simulation from G to H then $Z \subseteq Z_{\max}$. It can be shown that this simulation can be computed in time polynomial in the size of the two graphs G and H [HHK95]. In particular, this means that using Lemma 2.2 we can verify in polynomial time whether $x \in C^i$ holds. A similar characterization can be found for subsumption reasoning.

Lemma 2.3 (Characterizing subsumption in $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$)**.** *Let $C = (A_C, \mathcal{T}_C)$ and $D = (A_D, \mathcal{T}_D)$ be $\mathcal{EL}_{\text{gfp}}$-concept descriptions over the same signature. Then the following two statements are equivalent*

- $C \sqsubseteq D$
- *there is a simulation Z from the description graph of $\mathcal{T}_D$ to the description graph of $\mathcal{T}_C$ such that $(A_D, A_C) \in Z$.*

Obviously, the same holds if both C and D are $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions with $C \neq \bot$ and $D \neq \bot$. If $C = \bot$ or $D = \bot$ then deciding subsumption is trivial.

For matters of convenience we will often say that Z is *a simulation from the description $C = (A_C, \mathcal{T}_C)$ to the individual x in an interpretation i* meaning that Z is a simulation from the description graph of $\mathcal{T}_C$ to the description graph of i that contains (A_C, x). In other words Z is a simulation from C to x in i if

(S1′) $\text{names}_{\mathcal{T}_C}(B) \subseteq \text{names}_i(y)$ for all pairs $(B, y) \in Z$, and

(S2′) if $(B, y) \in Z$ and $B' \in \text{succ}^r_{\mathcal{T}_C}(B)$ then there is some $y' \in \text{succ}^r_i(y)$ such that $(B', y') \in Z$, and

(S3′) $(A_C, x) \in Z$.

Analogously, we say that Z is a simulation from $C = (A_C, \mathcal{T}_C)$ to $D = (A_D, \mathcal{T}_D)$ if

(S1″) $\text{names}_{\mathcal{T}_C}(B) \subseteq \text{names}_{\mathcal{T}_D}(y)$ for all pairs $(B, y) \in Z$, and

(S2″) if $(X, Y) \in Z$ and $X' \in \text{succ}^r_{\mathcal{T}_C}(X)$ then there is some $Y' \in \text{succ}^r_{\mathcal{T}_D}(Y)$ such that $(X', Y') \in Z$, and

(S3″) $(A_C, A_D) \in Z$.

In [Baa03a] Baader shows that the least common subsumers of $\mathcal{EL}_{\text{gfp}}$-concept descriptions always exist and provides a construction for them. The least common subsumer of a finite number of $\mathcal{EL}_{\text{gfp}}$-concept descriptions is obtained as their product. To formally define the product of $\mathcal{EL}_{\text{gfp}}$-concept descriptions we first define the product of $\mathcal{EL}$-description graphs.

Definition 2.25 (Product of $\mathcal{EL}$-description graphs)**.**
Let $G_1 = (V_1, E_1, L_1)$, …, $G_n = (V_n, E_n, L_n)$ be n $\mathcal{EL}$-description graphs. The *product* $G_1 \otimes \cdots \otimes G_n$ is the description graph $H = (W, F, M)$, where

- $W = V_1 \times \cdots \times V_2$,
- $F = \{((v_1, \ldots, v_n), r, (w_1, \ldots, w_n)) \mid r \in \mathcal{N}_R, \text{ and for all } 1 \leq k \leq n\colon (v_k, r, w_k) \in E_k\}$, and
- $M((v_1, \ldots, v_n)) = \bigcap_{1 \leq k \leq n} L_k(v_k)$.

The product of a finite set of $\mathcal{EL}_{\text{gfp}}$-concept descriptions is the concept description corresponding to the product of their $\mathcal{EL}$-description graphs.

Definition 2.26. Let $C_1 = (A_1, \mathcal{T}_1)$, ..., $C_n = (A_n, \mathcal{T}_n)$ be $\mathcal{EL}_{\text{gfp}}$-concept descriptions over the same signature. Let $G_1, \ldots, G_n$ be the description graphs of $T_1, \ldots, T_n$ be the description graphs of $\mathcal{T}_1, \ldots, \mathcal{T}_n$, respectively. The *product* $C_1 \otimes \cdots \otimes C_n$ is defined as the concept description $D = (A_D, \mathcal{T}_D)$ where

- $\mathcal{T}_D$ is the TBox that is obtained from the $\mathcal{EL}$-description graph $G_1 \otimes \cdots \otimes G_n$ using Definition 2.23, and
- $A_D = (A_1, \ldots, A_n)$.

Lemma 2.4. *Let $\{C_1, C_2, \ldots, C_n\}$ be a finite set of $\mathcal{EL}_{\text{gfp}}$-concept descriptions. Then $C_1 \otimes C_2 \otimes \cdots \otimes C_n$ is the least common subsumer of $\{C_1, C_2, \ldots, C_n\}$.*

This result enables us to compute least common subsumers for two $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions: Let C and D be two $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. Clearly, if $C = \bot$ then $\text{lcs}\{C, D\} = D$ and vice versa. The remaining case is that both C and D are $\mathcal{EL}_{\text{gfp}}$-concept descriptions. No $\mathcal{EL}_{\text{gfp}}$-concept description is subsumed by $\bot$ and therefore the least common subsumer of C and D in $\mathcal{EL}^{\bot}_{\text{gfp}}$ must be an $\mathcal{EL}_{\text{gfp}}$-concept description. Lemma 2.4 implies that the least $\mathcal{EL}_{\text{gfp}}$-concept description (with respect to $\sqsubseteq$) that subsumes both C and D is $C \otimes D$. Thus $\text{lcs}\{C, D\} = C \otimes D$.

Lemma 2.5. *Let C and D be $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. Then the least common subsumer of C and D exists and can be computed effectively.*

3 Formal Concept Analysis

Formal Concept Analysis [Wil82, GW97] is a field of discrete mathematics whose aim is to organize, analyze and process data and to discover structures within data. In this chapter we introduce the central notions of FCA such as formal contexts and formal concepts. Formal contexts are the underlying data structure of FCA. They can either be represented in the form of a cross table or in the form of a concept lattice. Concept lattices, which consist of formal concepts, present the information from the context in a structured way. We list existing algorithms for computing all formal concepts of a formal context. Among these algorithms the Next-Closure algorithm will be of particular interest in later chapters [Gan84]. We therefore explain it in detail in this section.

Implications between attributes are another way for structuring data in FCA. We introduce them in Section 3.3. The Duquenne-Guigues Base is a minimal set of implications from which all implications of a given formal context follow [GD86]. We show how it can be computed using Next-Closure. Implications also play an important role in the interactive (expert-assisted) knowledge acquisition formalism of FCA which is called Attribute Exploration. Attribute Exploration is the foundation of the knowledge base completion formalisms in Chapters 6 and 7. It is introduced in Section 3.4.

3.1 Formal Contexts and Formal Concepts

In Formal Concept Analysis data is usually represented in the form of a formal context. A formal concept contains information about so-called objects and their attributes. It specifies which attributes are associated to an object.

Definition 3.1 (Formal Context)**.** A *formal context* is a triple $\mathbb{K} = (G, M, I)$ where G and M are sets and $I \subseteq G \times M$ is a binary relation.

The elements of G are called *objects*, while the elements of M are called *attributes*. We say that an object g has attribute m if g and m are in relation I (denoted by gIm).

In the present work we assume that G and M are finite unless explicitly stated otherwise. Formal Contexts can be visualized in the form of cross tables like the one in Table 1.1. The columns of a cross table usually represent the attributes while the rows represent the objects. The relation I is represented using crosses, where a cross is placed in the row representing an object g and the column representing an attribute m iff the object g has the attribute m.

FCA provides two derivation operators in order to structure data in formal contexts. They can be used to derive the set of attributes that are common to a given set of objects, or, respectively, the set of objects that is common to a set of attributes.

Definition 3.2 (Derivation Operators)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context and $A \subseteq G$ be a set of objects. We define

$$A' = \{m \in M \mid \forall g \in A \colon gIm\},$$

i. e. A' is the set of all attributes that all objects in G share. Analogously, let $B \subseteq M$ be a set of attributes. We define

$$B' = \{g \in G \mid \forall m \in B \colon gIm\},$$

i. e. B' is the set of those objects that have all attributes from B.

The following properties of the derivation operators are quickly obtained from the definitions.

Lemma 3.1 (Properties of Derivation Operators)**.** *Let $\mathbb{K} = (G, M, I)$ be a formal context. Let $A_1, A_2 \subseteq G$ be sets of objects and let $B_1, B_2 \subseteq M$ be sets of attributes. Then the following statements hold.*

- $A_1 \subseteq A_2$ *implies* $A_2' \subseteq A_1'$
- $B_1 \subseteq B_2$ *implies* $B_2' \subseteq B_1'$
- $A_1 \subseteq A_1''$
- $B_1 \subseteq B_1''$

- $A_1' = A_1'''$
- $B_1' = B_1'''$
- $A_1 \subseteq B_1'$ *if and only if* $B_1 \subseteq A_1'$

Definition 3.3 (Formal Concept)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context. A *formal concept* is a pair (A, B) where $A \subseteq G$ is a set of objects, $B \subseteq M$ is a set of attributes and it holds that $A = B'$ and $B = A'$.

A is called the *concept extent* and B is called the *concept intent* of (A, B). B is called an *object intent* if $B = \{a\}'$ for some object $a \in M$.

We often say context, extent or intent instead of formal context, concept extent and concept intent. We will not abbreviate the expression "formal concept" since this might cause confusion with Description Logic concepts.

It is possible to define a partial order $\leq$ on the set of all formal concepts of a given context $\mathbb{K}$ by defining

$$(A_1, B_1) \leq (A_2, B_2) \text{ iff } A_1 \subseteq A_2.$$

It can be shown that the set of all formal concepts of $\mathbb{K}$ ordered by $\leq$ forms a complete lattice. This lattice is called the *concept lattice* of $\mathbb{K}$. FCA uses concept lattices as a tool for analyzing and visualizing a context.

An important application of formal concepts comes from data mining, or more precisely the computation of association rules. Efficient algorithms for mining association rules have been developed which make use of frequent closed itemsets, which are essentially intents of formal concepts with large extents [Stu02].

Both the visualization using concept lattices and the computation of closed itemsets require the enumeration of the formal concepts of a given context. It has therefore been a central topic in FCA research to find efficient enumeration algorithms. It has been shown in [Kuz04] that the number of formal concepts of a context can be exponential in the size of the context (i. e. $|G| \cdot |M|$). Therefore it is not possible to enumerate all formal concepts in polynomial time in the size of the context. The research has gone in the direction of finding algorithms whose runtime is bounded by a polynomial in the size of the input (the context) and output (the set of all formal concepts). Nowadays,

the most well-known algorithm for enumerating all intents is Ganter's Next-Closure [Gan84], which is presented in the following section. It is not only output-polynomial but even has polynomial delay, i. e. the time between the output of one formal concept and the next one is polynomial in the size of the context. Other polynomial delay algorithms can be found in [Kuz93, GMA95, NR99, STB+00]. For a detailed analysis of these algorithms we refer to [KO02].

3.2 Closure Operators and the Next-Closure Algorithm

Closure systems and closure operators are two closely related notions from order theory. The following definitions and results are common knowledge in order theory, cf. [DP02] for more details.

Definition 3.4 (Closure System)**.** Let X be a finite set, and $H \subseteq 2^X$. H is called a *closure system on* X if

- $X \in H$, and
- $U, V \in H$ implies $U \cap V \in H$.

The elements of a closure system are called closed sets.

Definition 3.5 (Closure Operator)**.** Let X be a finite set and $\mathfrak{cl}\colon 2^X \to 2^X$ a function mapping subsets of X to subsets of X. The function $\mathfrak{cl}$ is called a *closure operator on* X if

- $\mathfrak{cl}$ is idempotent, i. e. $\mathfrak{cl}(U) = \mathfrak{cl}(\mathfrak{cl}(U))$ for all $U \subseteq X$, and
- $\mathfrak{cl}$ is extensive, i. e. $U \subseteq \mathfrak{cl}(U)$ for all $U \subseteq X$, and
- $\mathfrak{cl}$ is order-preserving, i. e. $U \subseteq V$ implies $\mathfrak{cl}(U) \subseteq \mathfrak{cl}(V)$ for all $U, V \subseteq X$.

Every closure system gives rise to a closure operator and vice versa. Let X be a set and H a closure system on X. Then H gives rise to a closure operator

$$\mathfrak{cl}_H\colon U \mapsto \bigcap \{Y \in H \mid U \subseteq Y\}.$$

Since the intersection of closed sets is also closed, $\mathfrak{cl}_H$ maps a set U to the smallest closed set with respect to set inclusion that contains U. Conversely, a closure operator $\mathfrak{cl}$ gives rise to a closure system. Let $\mathfrak{cl}$ be a closure operator on a set X. Then

$$H_{\mathfrak{cl}} = \{\mathfrak{cl}(U) \mid U \subseteq X\}$$

is a closure system on X.

Given $\mathfrak{cl}$ and X, $H_{\mathfrak{cl}}$ can be computed by enumerating all subsets of X and applying $\mathfrak{cl}$. Obviously, this is not efficient as there are $2^{|X|}$ subsets of X, but the number of closed sets can be less than exponential in the size of X. Therefore the naive approach would have a worst-case runtime that is exponential not only in the size of the input $|X|$ but also in the size of the output $|H_{\mathfrak{cl}}|$. Next-Closure in its general form is an algorithm for enumerating all closed sets for a given closure operator $\mathfrak{cl}$ with polynomial delay. It makes use of a total order on the subsets of X called the *lectic order*. Let $<$ be a total order on the elements of X. Then we say that $A \subseteq X$ is *lectically smaller* than $B \subseteq X$ if the smallest element with respect to $<$ that distinguishes A and B is contained in B. Formally, we write

$$A < B :\Leftrightarrow \exists x \in B \setminus A \colon \forall y < x \colon (y \in A \Leftrightarrow y \in B).$$

Notice that the lectic order extends the subset order, i. e. $A \subsetneq B$ implies $A < B$. The relation $<$ is the union of the relations $<_x$ for all $x \in X$ where

$$A <_x B :\Leftrightarrow x \in B \setminus A \wedge \forall y < x \colon (y \in A \Leftrightarrow y \in B).$$

Lemma 3.2 (Next-Closure)**.** *Let X be a finite set. Let $\mathfrak{cl}$ be a closure operator on X and $A \subseteq X$ a set. If it exists the lectically smallest closed set that is lectically greater than A is*

$$\mathfrak{cl}(\{x\} \cup \{y \in A \mid y < x\})$$

where x is the greatest element of X for which $A <_x \mathfrak{cl}(\{x\} \cup \{y \in A \mid y < x\})$ holds. We call $\mathfrak{cl}(\{x\} \cup \{y \in A \mid y < x\})$ the next closure *of A (with respect to $\mathfrak{cl}$).*

Assuming that the closure operator $\mathfrak{cl}$ can be computed in polynomial time in the size of X and A Lemma 3.2 allows us to compute the lectically

next closed set in polynomial time (in the worst case we have to compute $\mathfrak{cl}(\{x\} \cup \{y \in A \mid y < x\})$ for all $|X|$ possible values of x). Next-Closure starts by computing the closure of the empty set $\emptyset$ and then successively computes the lectically next closed set. It terminates when it reaches the full set X (cf. Algorithm 1).

Algorithm 1 Next-Closure in its General Form

```
1: X, cl                                   {Input: set X and closure operator cl}
2: A_0 := cl(∅)                                  {lectically smallest closed set}
3: H_0 := {A_0}                          {initializing the set of closed sets}
4: i := 0
5: while A_i ≠ X do
6:     x_max := max{x ∈ X | A_i <_x cl({x} ∪ {y ∈ A_i | y < x})}
7:     A_{i+1} := cl({x_max} ∪ {y ∈ A_i | y < x_max})
8:     H_{i+1} := H_i ∪ {A_{i+1}}
9:     i := i + 1
10: end while
11: return H_i
```

Next-Closure can compute all formal concepts of a context. It is an easy consequence of Lemma 3.1 that we obtain a closure operator by consecutively applying the two derivation operators from FCA. Another consequence of Lemma 3.1 is the following result. It shows that the concept intents of a context $\mathbb{K}$ are exactly the sets of the form B'', where $B \subseteq M$, i. e. the concept intents are the closed sets of the closure operator $\cdot''$. Therefore, they can be enumerated using Next-Closure.

Lemma 3.3. *Let $\mathbb{K} = (G, M, I)$ be a context. Then every formal concept C of $\mathbb{K}$ is of the form*

$$C = (B', B'')$$

for some $B \subseteq M$.

The dual of this lemma is also true, i. e. every formal concept is of the form (A'', A') for some $A \subseteq G$. Algorithm 2 shows how Next-Closure can be used to find all formal concepts of a formal context. In Line 6 the lectically next closed set A_{i+1} is computed using Lemma 3.2, i. e. we first compute

$$x_{\max} := \max\{x \in M \mid A_i <_x (\{x\} \cup \{y \in A_i \mid y < x\})''\}$$

and obtain

$$A_{i+1} := (\{x_{\max}\} \cup \{y \in A_i \mid y < x_{\max}\})''.$$

Algorithm 2 Next-Closure for Enumerating all Formal Concepts

```
1: 𝕂 := (G, M, I)                          {Input: formal context 𝕂}
2: A_0 := ∅''                              {lectically smallest concept intent}
3: C_0 := {(A_0', A_0)}                    {initializing the set of formal concepts}
4: i := 0
5: while A_i ≠ M do
6:    A_{i+1} := lectically smallest subset of M that is
      – lectically greater than A_i, and
      – closed with respect to ·''
7:    C_{i+1} := C_i ∪ {(A_{i+1}', A_{i+1})}
8:    i := i + 1
9: end while
10: return C_i
```

3.3 Implications and the Duquenne-Guigues Base

Concept lattices are a method to structure the information of a context. Another way to analyze contexts are implications. Implications between attributes are statements of the form "All objects that have all attributes from the set A also have all attributes from the set B".

Definition 3.6 (Implications between Sets of Attributes)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context. An *implication* is a pair (A, B) where $A, B \subseteq M$. Usually, we denote the implication by $A \to B$.

We say that *an implication $A \to B$ holds in the context* $\mathbb{K}$ if all objects that have all attributes from A also have all attributes from B, i. e. if $A' \subseteq B'$.

As an example, let us have a look at the context $\mathbb{K}_S$ from Table 1.1. There are two objects that have the attribute Father, namely Homer

and Kirk, and both of them also have the attribute Parent. Therefore $\{\mathsf{Father}\} \to \{\mathsf{Parent}\}$ is an implication that holds in $\mathbb{K}_S$.

Implications are an intuitive way to reveal dependencies that exist within the data. However, computing all implications that hold in a given context $\mathbb{K}$ is not practical. Instead we are interested in a small set of implications from which all other implications of the context follow.

Definition 3.7 (Implicational Base)**.** Let M be a set of attributes and let $A, B \subseteq M$ be two subsets. A set $C \subseteq M$ is said to *respect* the implication $A \to B$ if $A \subseteq C$ implies $B \subseteq C$.

An implication $A \to B$ *follows* from a set of implications $\mathcal{L}$ if every subset $C \subseteq M$ that respects all implications from $\mathcal{L}$ also respects $A \to B$.

Let $\mathbb{K} = (G, M, I)$ be a context. We say that the set of implications $\mathcal{L}$ is a *(implicational) base* of $\mathbb{K}$ if

- $\mathcal{L}$ is *sound* for $\mathbb{K}$, i. e. every implication from $\mathcal{L}$ holds in $\mathbb{K}$, and
- $\mathcal{L}$ is *complete* for $\mathbb{K}$, i. e. every implication that holds in $\mathbb{K}$ follows from $\mathcal{L}$. [1]

It is a well-known fact from Formal Concept Analysis that $A \to B$ *follows* from a set of implications $\mathcal{L}$ iff it is derivable from $\mathcal{L}$ via the Armstrong rules [Arm74]. In general there exists more than one base for a given context $\mathbb{K}$. The simplest implicational base of a context is the set of all implications that hold in the context, which is usually too large to be practical. Usually we are interested in small bases or, if possible, bases with minimal cardinality.

Lemma 3.4. *Let $\mathbb{K} = (G, M, I)$ be a formal context and $A \to B$ an implication that holds in $\mathbb{K}$. Then $A \to B$ follows from $\{A \to A''\}$. In particular this shows that the set $\{A \to A'' \mid A \subseteq M\}$ is an implicational base of $\mathbb{K}$.*

Lemma 3.4 shows that to compute a base of minimal cardinality it suffices to use implications whose right-hand side is a concept intent. It is the reason why the following section focusses only on finding "good" left-hand sides.

[1] Most authors additionally require that the base be irredundant. Thus Definition 3.7 is more general than the usual definition of an implicational base in FCA.

Pseudo-Intents and the Duquenne-Guigues Base

In [GD86] Duquenne and Guigues present a construction for a natural base for every context. This base is not only irredundant, but also has minimal cardinality among all bases of this context. It is called the *Duquenne-Guigues Base* or the *stem base* of this context. It is based on the notion of pseudo-intents which is defined inductively.

Definition 3.8 (Pseudo-Intent)**.** Let $\mathbb{K} = (G, M, I)$ be a context. A set $P \subseteq M$ is called a *pseudo-intent of* $\mathbb{K}$ if

- P is not an intent, and
- for all pseudo-intents $Q \subsetneq P$ it holds that $Q'' \subseteq P$.

There exists an equivalent non-recursive definition for pseudo-intents that can be found in [Kuz06] amongst others.

Theorem 3.5 (Duquenne-Guigues Base)**.** *Let $\mathbb{K} = (G, M, I)$ be a context. The set of implications*

$$\mathcal{DG}_{\mathbb{K}} = \{P \to P'' \mid P \text{ is a pseudo-intent of } \mathbb{K}\}$$

is a base of minimal cardinality for $\mathbb{K}$. $\mathcal{L}$ is called the Duquenne-Guigues Base *of $\mathbb{K}$.*

Example 3.1. In the context $\mathbb{K}_S$ from Table 1.1 the pseudo-intents are the sets $\{\mathsf{Father}\}$, $\{\mathsf{Mother}\}$, $\{\mathsf{Female}, \mathsf{Parent}\}$, $\{\mathsf{Male}, \mathsf{Parent}\}$, and $\{\mathsf{Female}, \mathsf{Male}\}$. This yields the following Duquenne-Guigues Base

$$\begin{aligned}
\{\mathsf{Father}\} &\to \{\mathsf{Male}, \mathsf{Father}, \mathsf{Parent}\} \\
\{\mathsf{Mother}\} &\to \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\} \\
\{\mathsf{Male}, \mathsf{Parent}\} &\to \{\mathsf{Male}, \mathsf{Father}, \mathsf{Parent}\} \\
\{\mathsf{Female}, \mathsf{Parent}\} &\to \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\} \\
\{\mathsf{Female}, \mathsf{Male}\} &\to \{\mathsf{Female}, \mathsf{Male}, \mathsf{Mother}, \mathsf{Father}, \mathsf{Parent}\}.
\end{aligned}$$

Using an FCA implication it is not possible to express that an object that has both attributes Female and Male does not exist. The last implication which states that such an object has all possible attributes is the closest approximation to such a statement.

An algorithm for computing the Duquenne-Guigues Base (or the set of all pseudo-intents) of a context is based on the following observation.

Lemma 3.6 (Closure System of Intents and Pseudo-Intents)**.** *Let* $\mathbb{K} = (G, M, I)$ *be a context. The set of all intents and pseudo-intents of* $\mathbb{K}$ *is a closure system on* M*.*

Because the set of all intents and pseudo-intents forms a closure system it can be computed using Next-Closure. To this purpose we need to identify the corresponding closure operator and an effective way to compute it. In order to identify the closure operator we define the implicational closure of a set of attributes.

Definition 3.9 (Implicational Closure)**.** Let $A \subseteq M$ be a set of attributes and $\mathcal{L}$ a set of implications. Then the *implicational closure* of A with respect to $\mathcal{L}$ is the smallest set B, $A \subseteq B \subseteq M$, that respects all implications from $\mathcal{L}$. We denote the implicational closure of A with respect to $\mathcal{L}$ by $\mathcal{L}(A)$.

One can show that the implicational closure $\mathcal{L}(\cdot)$ is a closure operator. The closed sets with respect to $\mathcal{L}(\cdot)$ are exactly those sets that respect all implications from $\mathcal{L}$. $\mathcal{L}(A)$ can computed in linear time in the size of A and $\mathcal{L}$. This can be proven in different ways, for example by interpreting FCA implications as propositional Horn clauses and using results for deciding satisfiability of propositional Horn clauses [DG84].

Definition 3.10 (Pseudo-Closure)**.** Let $\mathbb{K} = (G, M, I)$ be a context. Let $A \subseteq M$ be a set of attributes. The pseudo-closure of A is the smallest set B, $A \subseteq B \subseteq M$, such that

$$L \subsetneq B \text{ implies } R \subseteq B$$

for all implications $L \to R$ from $\mathcal{DG}_{\mathbb{K}}$. We denote the pseudo-closure by $\mathcal{DG}^*_{\mathbb{K}}(A)$.

It can be shown that $\mathcal{DG}^*_{\mathbb{K}}$ is the closure operator that generates the closure system of all intents and pseudo-intents. Notice that if $P = \mathcal{DG}^*_{\mathbb{K}}(A)$ for some $A \subseteq M$ then $P = \mathcal{L}(A)$ where $\mathcal{L} = \{L \to R \in \mathcal{DG}_{\mathbb{K}} \mid L \subsetneq P\}$ is the set of all implications from $\mathcal{DG}_{\mathbb{K}}$ whose left-hand side is strictly contained in P. Therefore, when we want to compute the lectically next intent or pseudo-intent for a given set A it is sufficient

to know those implications from $\mathcal{DG}_{\mathbb{K}}$ whose left-hand side is strictly contained in A. Remember that the lectic order extends the subset order. Since Next-Closure computes the intents and the pseudo-intents in the lectic order these implications are within the already computed part of the Duquenne-Guigues Base, which gives us an effective way to compute the pseudo-closure of a set of attributes. The resulting algorithm is presented in Algorithm 3. It computes not only the intents and pseudo-intents but also the Duquenne-Guigues Base.

Kuznetsov has shown that the total number of pseudo-intents of a context $\mathbb{K} = (G, M, I)$ can be exponential in $|G| \cdot |M|$. Of course, this means that the size of the Duquenne-Guigues base can also become exponentially large in $|G| \cdot |M|$. In Line 7 computing the pseudo-closure by computing the implicational closure with respect to the previously found implications is not necessarily polynomial in the size of the input ($|G| \cdot |M|$) but polynomial in the size of the input and the previously computed output ($|G| \cdot |M|$ and $|\mathcal{L}_{i+1}|$). Hence Algorithm 3 computes the set of all intents and pseudo-intents with *incremental* polynomial delay, and thus in output polynomial time.

Algorithm 3 Computing the Duquenne-Guigues-Base of a Context

```
1:  𝕂                                              {Input: formal context 𝕂}
2:  P_0 := ∅ {∅ is always the lectically smallest intent or pseudo-intent}
3:  ℒ_0 := ∅                                {initializing the set of implications}
4:  i := 0
5:  while P_i ≠ M do
6:    if P_i ≠ P_i'' then
7:      ℒ_{i+1} := ℒ_i ∪ {P_i → P_i''}                    {P_i is a pseudo-intent}
8:    end if
9:    P_{i+1} := lectically smallest subset of M that is
        – lectically greater than P_i, and
        – respects all implications from ℒ_{i+1}
10:   i := i + 1
11: end while
12: return ℒ_i
```

Later we shall discuss applications of FCA in DL. These do not make use of concept intents but only of pseudo-intents. For computing only

pseudo-intents Next-Closure is not efficient, since it will also compute the intents. The number of intents may be exponential in the number of pseudo-intents. To this day, it is an open problem whether the set of pseudo-intents can be enumerated in output polynomial time. In [Ser09b] a connection between this enumeration problem and a problem from hypergraph theory, namely the transversal hypergraph problem, is shown. The author of this thesis has proven that pseudo-intents cannot be enumerated in the lectic order with polynomial delay unless P = NP [Dis10b, DS11]. A closely related complexity problem is the problem of verifying whether a given set of attributes is a pseudo-intent. This problem has been known to be in coNP [Kuz04, KO08] and has recently been shown to be coNP-complete in [BK10].

3.4 Attribute Exploration

In practice it often happens that a context is only partially available. Computing the Duquenne-Guigues Base of such a partially available context is problematic. An implication that is valid in a smaller subcontext is not necessarily valid in the full context, because that full context may contain a counterexample that is not present in the subcontext. As an example consider the context $\mathbb{K}_S$ from Table 1.1 as the full context and the context $\mathbb{K}_p$ from Table 3.1 as the subcontext. The implication $\{\mathsf{Female}\} \to \{\mathsf{Mother}\}$ holds in $\mathbb{K}_p$ but not in $\mathbb{K}_S$ because the counterexample Lisa is missing in $\mathbb{K}_p$.

Attribute Exploration is an interactive procedure for completing formal contexts while at the same time computing the Duquenne-Guigues Base for the complete context [Gan84]. It takes as input an incomplete formal context and obtains more information by querying a human expert. The expert is assumed to have knowledge about the complete context. The Attribute Exploration algorithm consecutively computes implications and presents them to the expert. For each implication the expert has the choice to either accept or to reject it. If she accepts it, it will be added to the implication base that is being computed. If she rejects it she is asked to provide a counterexample in the form of an object and its object intent. The complete algorithm is described in Algorithm 4. It is a modification of Next-Closure for computing the Duquenne-Guigues Base. Remarkably, Lemma 3.7 shows that the

previously computed implications remain valid and remain part of the Duquenne-Guigues Base even when the expert adds a counterexample.

Algorithm 4 Attribute Exploration

$\mathbb{K}_0 := (G, M_0, I_0)$ {Input: initial formal context $\mathbb{K}_0$}
$P_0 := \emptyset$ {$\emptyset$ is always the lectically smallest intent or pseudo-intent}
$\mathcal{L}_0 := \emptyset$ {initializing the set of implications}
$i := 0$; $j := 0$
while $P_i \neq M$ **do**
  while $P_i \neq P_i''$ and the expert rejects $P_i \to P_i''$ **do**
    Ask expert for a new object g and its object intent g'
    $M_{j+1} := M_j \cup \{g\}$
    $I_{j+1} := I_j \cup \{(g, m) \mid m \in g'\}$
    $\mathbb{K}_{j+1} := (G, M_{i+1}, I_{i+1})$
    $j := j + 1$
  end while
  if $P_i \neq P_i''$ **then**
    $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \to P_i''\}$
  end if
  $P_{i+1} :=$ lectically smallest subset of M that is
  – lectically greater than P_i, and
  – respects all implications from $\mathcal{L}_{i+1}$
  $i := i + 1$
end while
return $\mathcal{L}_i$

Lemma 3.7. *Let $\mathbb{K} = (G, M, I)$ and $\mathbb{K}' = (G', M, I')$ be formal contexts. We say that $\mathbb{K}'$* extends *$\mathbb{K}$ if $G \subseteq G'$ and $I' \setminus I \subseteq (G' \setminus G) \times M$, i. e. for all objects from G the object intents remain unchanged.*

Let $\mathbb{K}$ be a context and let P_1, P_2, ..., P_k be its k lectically first pseudo-intents. Let $\mathbb{K}'$ be a context that extends $\mathbb{K}$ and where all implications $P_\ell \to P_\ell''$, $\ell \in \{1, \ldots, k\}$ hold in $\mathbb{K}'$. Then P_1, P_2, ..., P_k are the lectically first k pseudo-intents of $\mathbb{K}'$.

Example 3.2. Assume that instead of the complete context $\mathbb{K}_S$ from Table 1.1 only an incomplete version available (Table 3.1). The task is to find all the relevant implications, i. e. the implications from the

Table 3.1: A Formal Context $\mathbb{K}_p$

	Female	Male	Mother	Father	Parent
Kirk		×		×	×
Luann	×		×		×
Milhouse		×			

Duquenne-Guigues Base of the complete context $\mathbb{K}_S$, by interacting with a human expert.

The first implication that Algorithm 4 finds is

$$\{\mathsf{Father}\} \to \{\mathsf{Male}, \mathsf{Father}, \mathsf{Parent}\}.$$

Since this is true in the full context $\mathbb{K}_S$, the expert accepts this implication. Likewise, the implications

$$\{\mathsf{Mother}\} \to \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\}, \text{ and}$$
$$\{\mathsf{Male}, \mathsf{Parent}\} \to \{\mathsf{Male}, \mathsf{Father}, \mathsf{Parent}\}$$

are accepted and added to the implication base. The fourth implication is

$$\{\mathsf{Female}\} \to \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\}.$$

This does not hold in $\mathbb{K}_S$ since Lisa is Female, but she is not a Parent. Hence, the expert rejects the implication and provides the counterexample Lisa. The last two implication are accepted and no counterexample is provided. After these last questions the Attribute Exploration terminates:

$$\{\mathsf{Female}, \mathsf{Parent}\} \to \{\mathsf{Female}, \mathsf{Mother}, \mathsf{Parent}\}$$
$$\{\mathsf{Female}, \mathsf{Male}\} \to \{\mathsf{Female}, \mathsf{Male}, \mathsf{Mother}, \mathsf{Father}, \mathsf{Parent}\}.$$

Background Knowledge

In the following chapters we sometimes deal with a situation where certain dependencies between attributes are known in advance, e. g. because the attributes are DL concept descriptions and some dependencies

are obtained using DL reasoning. This setting has first been examined in [Stu96a] and [Gan99]. We follow the notation from [Stu96a]. We assume that we are given a formal context $\mathbb{K} = (G, M, I)$ and a set of previously known implications $\mathcal{S}$. The goal is to find a minimal set of implications $\mathcal{L}$ such that

- all implications from $\mathcal{L}$ hold in $\mathbb{K}$, and
- all implications that hold in $\mathbb{K}$ follow from $\mathcal{L} \cup \mathcal{S}$.

We call such a set an *$\mathcal{S}$-base* of $\mathbb{K}$. An $\mathcal{S}$-base with minimal cardinality can be constructed in analogy to the Duquenne-Guigues Base.

Definition 3.11 ($\mathcal{S}$-Pseudo-Intent)**.** Let $\mathbb{K} = (G, M, I)$ be a context and $\mathcal{S}$ be a set of implications holding in $\mathbb{K}$. A set $P \subseteq M$ is called an *$\mathcal{S}$-pseudo-intent* of $\mathbb{K}$ if

- P is not an intent, and
- P respects all implications from $\mathcal{S}$, and
- for all $\mathcal{S}$-pseudo-intents $Q \subsetneq P$ it holds that $Q'' \subseteq P$.

We call

$$\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}} = \{P \to P'' \mid P \text{ is an } \mathcal{S} \text{ pseudo-intent of } \mathbb{K}\}.$$

the $\mathcal{S}$-Duquenne-Guigues Base of $\mathbb{K}$

Theorem 3.8. *Let $\mathbb{K}$ be a context and $\mathcal{S}$ a set of implications that hold in $\mathbb{K}$. Then the $\mathcal{S}$-Duquenne-Guigues Base $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$ is an $\mathcal{S}$-base of $\mathbb{K}$. It has minimal cardinality among all $\mathcal{S}$-bases of $\mathbb{K}$.*

Proof. The first statement that $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$ is $\mathcal{S}$-base of $\mathbb{K}$ is proved in [Stu96a]. However, in [Stu96a] Stumme does not prove minimal cardinality, but only the weaker result that $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$ is irredundant. We therefore provide the proof for minimal cardinality here.

Let $\mathcal{L}$ be an $\mathcal{S}$-base for $\mathbb{K}$. Lemma 3.4 allows us to transform $\mathcal{L}$ into an $\mathcal{S}$-base $\mathcal{L}'$ of the same size or smaller size by defining

$$\mathcal{L}' = \{U \to U'' \mid U \text{ occurs as a left-hand side in } \mathcal{L}\}.$$

Let P and Q be two $\mathcal{S}$-pseudo-intents. It holds that $P \neq P''$ and $Q \neq Q''$. P and Q respect all implications from $\mathcal{S}$ and $\mathcal{L}' \cup \mathcal{S}$ is complete for $\mathbb{K}$. Therefore there must be implications $U_P \to U_P''$ and $U_Q \to U_Q''$ in $\mathcal{L}'$ such that $U_P \subseteq P$, $U_Q \subseteq Q$ and $U_P'' \not\subseteq P$ and $U_Q'' \not\subseteq Q$. We show that $U_P = U_Q$ implies $P = Q$.

Assume that $U_P = U_Q = U$. Then U satisfies $U \subseteq P \cap Q$. From Lemma 3.1 it follows that $U'' \subseteq (P \cap Q)''$. $U_P'' = U'' \not\subseteq P$ and $U'' \subseteq (P \cap Q)''$ implies

$$(P \cap Q)'' \not\subseteq P \tag{3.1}$$

and analogously $(P \cap Q)'' \not\subseteq Q$. Therefore in particular $(P \cap Q)'' \not\subseteq P \cap Q$ and hence

$$P \cap Q \neq (P \cap Q)'' \tag{3.2}$$

holds. Since P and Q respect all implications from $\mathcal{S}$ the set $P \cap Q$ must also respect all implications from $\mathcal{S}$. Because of (3.2) and because $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}} \cup \mathcal{S}$ is complete for $\mathbb{K}$ there must be some $\mathcal{S}$-pseudo-intent R such that $R \subseteq P \cap Q$ and $R'' \not\subseteq P \cap Q$. Thus either $R'' \not\subseteq P$ or $R'' \not\subseteq Q$. Without loss of generality let $R'' \not\subseteq Q$. Since both Q and R are $\mathcal{S}$-pseudo-intents we obtain that $R = Q$ and thus $Q \subseteq P \cap Q$. This implies $Q = P \cap Q$ and therefore $Q \subseteq P$. From (3.1) we obtain that $Q'' \not\subseteq P$. Since both P and Q are $\mathcal{S}$-pseudo-intents it follows that $P = Q$.

We have seen that for every $\mathcal{S}$-pseudo-intent P there is at least one implication $U_P \to U_P''$ in $\mathcal{L}'$ such that $U_P \subseteq P$ and $U_P'' \not\subseteq P$. Furthermore, we have seen that if P and Q are distinct $\mathcal{S}$-pseudo-intents then $U_P \neq U_Q$ holds. This shows that $\mathcal{L}'$ and therefore also $\mathcal{L}$ contains at least as many implications as there are $\mathcal{S}$-pseudo-intents in $\mathbb{K}$. □

We call $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$ the *$\mathcal{S}$-Duquenne-Guigues Base* of $\mathbb{K}$. In order to compute $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$ we use the same ideas as for computing the Duquenne-Guigues Base for a context without background knowledge. It can be shown that the set of all $\mathcal{S}$-pseudo-intents and all intents of $\mathbb{K}$ forms a closure system. Therefore Next-Closure can be used to compute them. In analogy to Algorithm 3 the corresponding closure operator can be computed as the implicational closure with respect to $\mathcal{S}$ and the already obtained implications from $\mathcal{DG}_{\mathbb{K}}^{\mathcal{S}}$. Algorithm 5 shows a modified version of Algorithm 3 that computes all intents and $\mathcal{S}$-pseudo-intents for a given context. Using similar modifications Algorithm 4 can be modified to

Algorithm 5 Computing all Intents and $\mathcal{S}$-Pseudo-Intents of a Formal Context

1: $\mathbb{K}$, $\mathcal{S}$ {Input: formal context $\mathbb{K}$ and background implications $\mathcal{S}$}
2: $P_0 := \mathcal{S}(\emptyset)$ {lectically smallest intent or $\mathcal{S}$-pseudo-intent}
3: $H_0 := \{P_0\}$ {initializing the set of intents and $\mathcal{S}$-pseudo-intents}
4: $\mathcal{L}_0 := \emptyset$ {initializing the set of implications}
5: $i := 0$
6: **while** $P_i \neq M$ **do**
7: **if** $P_i \neq P_i''$ **then**
8: $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \to P_i''\}$ {P_i is an $\mathcal{S}$-pseudo-intent}
9: **end if**
10: P_{i+1} := lectically smallest subset of M that is
– lectically greater than P_i, and
– respects all implications from $\mathcal{L}_{i+1} \cup \mathcal{S}_{i+1}$
11: $H_{i+1} := H_i \cup \{P_{i+1}\}$
12: $i := i + 1$
13: **end while**
14: **return** $\mathcal{L}_i$

obtain an Attribute Exploration algorithm that can handle background knowledge.

4 General Frameworks for Combining FCA and DL

When we compare FCA and DL we realize that both theories have certain shortcomings with respect to the other one. The two most obvious shortcomings of FCA compared to DL are the lack of roles and the fact that it is impossible to construct new attributes from existing ones. FCA's advantage over DL is that it provides methods to extract dependencies between attributes in the form of implications. This section focusses on general approaches for combining FCA and DL. The idea is that by combining FCA and DL one can overcome each theory's shortcomings and obtain the best of both worlds.

In the past there have been several approaches to combine FCA and DL. A simple way of transporting DLs expressivity to a formal context is to use contexts whose attributes are DL concept descriptions. In this work we call a context that is obtained in this way an *induced context*. It has been used under different names and in slightly different versions in almost all previous works that combine FCA and DL [BGSS07, Rud04, Rud06, Pre00, FR04]. Apart from this there are many approaches to deal with roles (or relations) in FCA, most of which are not tailored towards DL. One way to allow relations in FCA are so-called *power context families* where objects are n-tuples over a basic set G_0 and n-ary relations correspond to attribute extents [PW99, Rud06]. Other approaches include *Relational Concept Analysis* by Priss [Pri98] and transition contexts by Wollbold et al. [WGG08]. All of these approaches use FCA as the starting point and add a DL flavor to it. In the first part of this chapter we use the opposite approach where we start with normal DL and introduce a new non-standard reasoning service that has similar properties as an FCA derivation operator. We call this new non-standard reasoning service the *model-based most specific concept*.

This section is divided into two parts: the first is devoted to model-based most specific concepts while the second is devoted to induced

contexts. In the first part we formally introduce model-based most specific concepts and show that they have several properties that make them similar to an FCA derivation operator. Unfortunately, model-based most specific concepts do not exist in all Description Logics. We prove that they always exist in the logics $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$. In the second part induced contexts are introduced formally, and some technical results are shown.

4.1 Model-Based Most Specific Concepts

When we compare the data structures of a formal context $\mathbb{K} = (G, M, I)$ and a DL-interpretation $i = (\Delta_i, \cdot^i)$ we can notice several parallels. An obvious parallel exists between the set of objects G of a context and the set individuals Δ_i of an interpretation. In both structures we can generate descriptions for subsets of G or Δ_i and both structures provide a mapping that returns the set of objects or individuals that satisfy a description. In the formal context such a description is a set of attributes $A \subseteq M$ and the mapping that interprets a description is the derivation operator $\cdot'$. In the DL world we have complex, inductively defined concept descriptions and the interpretation provides the mapping $\cdot^i$ to interpret them. Last but not least, there are parallels between implications in formal contexts, i. e. ordered pairs of sets of attributes, and GCIs in Description Logics, i. e. ordered pairs of concept descriptions.

These parallels between FCA implications on the one hand and GCIs on the other hand suggest that it might be possible to transfer the ideas that are used in the construction of the Duquenne-Guigues Base to the world of description logics. One quickly realizes that while there is an equivalent to the derivation operator $\cdot' : 2^M \to 2^G$, namely the interpretation function $\cdot^i$, there is no such equivalent for the dual operator $\cdot' : 2^G \to 2^M$. Our search for a reasoning service with similar properties as the second derivation operator leads us to model-based most specific concepts.

4.1.1 General Definition

We introduce model-based most specific concepts for a generic DL and show that they have similar properties as the derivation operator $\cdot'$:

$2^G \to 2^M$. In particular, Lemma 3.1 and Lemma 3.4 have equivalents in the DL setting (Lemma 4.1 and Lemma 4.3).

Definition 4.1 (Model-Based Most Specific Concept)**.** Let $\mathcal{L}$ be the set of all possible concept descriptions from some DL language over a fixed signature. Let $i = (\Delta_i, \cdot^i)$ be an interpretation and $X \subseteq \Delta_i$. Then $C \in \mathcal{L}$ is the *model-based most specific concept (mmsc) of* X if

(M1) $X \subseteq C^i$, and

(M2) every other concept description $D \in \mathcal{L}$ satisfying $X \subseteq D^i$ also satisfies $C \sqsubseteq D$.

(M2) implies in particular that model-based most specific concepts are unique up to equivalence. This justifies calling a concept description C *the* model-based most specific concept. We denote the model-based most specific concept of X by X^i. Model-based most specific concepts should not be confused with most specific concepts for ABox-individuals (cf. Definition 2.13). While the idea behind the two notions is similar the first refers to individuals from an interpretation, which have closed world semantics, while the latter refers to ABox-individuals, which have an open-world semantics. In the following section we shall see that model-based most specific concepts need not exist for all DL languages. For the rest of this section, we assume that we are dealing with a language where model-based most specific concepts always exist. We show that they have properties that ressemble those of the derivation operator from FCA.

Lemma 4.1. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation, $X, Y \subseteq \Delta_i$ sets of individuals, and C, D concept descriptions. Then the following statements hold.*

1. *$X \subseteq Y$ implies $X^i \sqsubseteq Y^i$*
2. *$C \sqsubseteq D$ implies $C^i \subseteq D^i$*
3. *$X \subseteq (X^i)^i$*
4. *$(C^i)^i \sqsubseteq C$*
5. *$X^i \equiv ((X^i)^i)^i$*

6. $C^i = ((C^i)^i)^i$
7. $X \subseteq C^i \Leftrightarrow X^i \sqsubseteq C$

In particular this shows that the interpretation function and the model-based most specific concept form a monotone Galois-connection.

Proof. (1) Assume that $X \subseteq Y$ holds. (M1) states in particular that $Y \subseteq (Y^i)^i$ and therefore $X \subseteq (Y^i)^i$. By definition X^i is the least concept description with $X \subseteq (X^i)^i$. (M2) and $X \subseteq (Y^i)^i$ imply that $X^i \sqsubseteq Y^i$ holds. *(2)* If $C \sqsubseteq D$ then Definition 2.9 says that $C^j \subseteq D^j$ holds for all interpretations j and thus in particular i satisfies $C^i \subseteq D^i$.

(3) The claim follows trivially from (M1). *(4)* Define $Y = C^i$. Then in particular $Y \subseteq C^i$ holds. By Definition 4.1 Y^i is the least concept description whose extension contains Y and therefore (M2) and $Y \subseteq C^i$ imply that $Y^i \sqsubseteq C$ holds. Because $Y = C^i$ it follows that $(C^i)^i \sqsubseteq C$.

Together, (1), (2), (3), and (4) show that the interpretation function and the model-based most specific concept form a monoton Galois-connection (e. g. [EKMS93]). (5) and (6) hold for every Galois-connection and (7) is simply the definition of a monotone Galois-connection. □

Notice that Lemma 4.1 corresponds almost exactly to Lemma 3.1. The main difference is that in Lemma 4.1 the direction of the order relation $\sqsubseteq$ is reversed compared to the subset order on sets of attributes n Lemma 3.1.

In the following we abbreviate the term $(C^i)^i$, where C is a concept description, by C^{ii}. Lemma 4.1 (6) proves a kind of idempotency for the combined operator $\cdot^{ii}$. According to the following result, this kind of idempotency holds, even when the operator $\cdot^{ii}$ occurs behind a quantifier.

Lemma 4.2. *Let $\mathcal{L}$ be a Description Logic that allows for conjunction and existential restrictions. Let C and D be concept descriptions from $\mathcal{L}$ and let $i = (\Delta_i, \cdot^i)$ be an interpretation. Then the following statements hold.*

1. $(C^{ii} \sqcap D)^i = (C \sqcap D)^i$
2. $(\exists r.C^{ii})^i = (\exists r.C)^i$

Proof. (1) From Lemma 4.1 (6) and the definition of conjunction in DL we obtain

$$(C^{ii} \sqcap D)^i = (C^{ii})^i \cap D^i = C^i \cap D^i = (C \sqcap D)^i.$$

(2) Let $x \in \Delta_i$ be an individual. The following equivalences follow from the definition of existential restrictions and from Lemma 4.1 (6).

$$\begin{aligned} & x \in (\exists r.C^{ii})^i \\ \Leftrightarrow & \exists y \in (C^{ii})^i : (x, y) \in r^i \\ \Leftrightarrow & \exists y \in C^i : (x, y) \in r^i \\ \Leftrightarrow & x \in (\exists r.C)^i. \end{aligned}$$

Hence, $(\exists r.C^{ii})^i = (\exists r.C)^i$ holds. □

We have already mentioned the similarities between Lemma 4.1 and Lemma 3.1. A similar correspondence can be obtained for Lemma 3.4.

Lemma 4.3. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation and let $C \sqsubseteq D$ be a GCI that holds in i. Then $C \sqsubseteq C^{ii}$ holds in i and $C \sqsubseteq D$ follows from $\{C \sqsubseteq C^{ii}\}$.*

Proof. By Lemma 4.1 (6) it holds that $C^i = (C^{ii})^i$, i. e. $C \sqsubseteq C^{ii}$ holds in i. We have assumed that $C \sqsubseteq D$ holds in i and therefore $C^i \subseteq D^i$ holds. Lemma 4.1 (7) implies $(C^i)^i \sqsubseteq D$. Let $j = (\Delta_j, \cdot^j)$ be any interpretation in which $C \sqsubseteq C^{ii}$ holds, i. e. j satisfies $C^j \subseteq (C^{ii})^j$. Using Lemma 4.1 (7) we obtain $(C^j)^j \sqsubseteq C^{ii}$. We have already shown $C^{ii} \sqsubseteq D$ and thus $(C^j)^j \sqsubseteq D$ follows. Lemma 4.1 (7) yields $C^j \subseteq D^j$, and hence $C \sqsubseteq D$ holds in j. Since j was an arbitrary interpretation in which $C \sqsubseteq C^{ii}$ holds, we have shown that $C \sqsubseteq D$ follows from $\{C \sqsubseteq C^{ii}\}$. □

4.1.2 Existence in the $\mathcal{EL}$-family

In the previous section we have shown that model-based most specific concepts – provided that they exist – have several desirable properties that ressemble the derivation operators from FCA. In this section we show that unfortunately they need not exist in $\mathcal{EL}$ or $\mathcal{EL}^\bot$. The reason for this is that $\mathcal{EL}$ and $\mathcal{EL}^\bot$ cannot describe cyclic dependencies – a problem that can be overcome by using $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^\bot_{\text{gfp}}$. In the later

parts of this section we show that model-based most specific concepts do exist in $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$.

To see that in general model-based most specific concepts do not exist in $\mathcal{EL}$ or $\mathcal{EL}^{\bot}$ let us look at the signature $(\mathcal{N}_C, \mathcal{N}_R) = (\{\mathsf{Male}, \mathsf{Female}\}, \{\mathsf{marriedTo}\})$ and the interpretation $i = (\{\mathsf{Homer}, \mathsf{Marge}\}, \cdot^i)$ with

$$\begin{aligned} \mathsf{Male}^i &= \{\mathsf{Homer}\} \\ \mathsf{Female}^i &= \{\mathsf{Marge}\} \\ \mathsf{marriedTo}^i &= \{(\mathsf{Homer}, \mathsf{Marge}), (\mathsf{Marge}, \mathsf{Homer})\} \end{aligned} \tag{4.1}$$

from Figure 2.2 again. Define

$$C_k = \underbrace{\exists \mathsf{marriedTo} \ldots \exists \mathsf{marriedTo}.}_{k \text{ times}} \top.$$

For all $k \in \mathbb{N}$ the concept C_k satisfies $C_k^i = \{\mathsf{Homer}, \mathsf{Marge}\}$. Suppose that a model-based most specific concept D of $\{\mathsf{Homer}, \mathsf{Marge}\}$ exists. Then D must satisfy $D \sqsubseteq C_k$ for all $k \in \mathbb{N}$. It is easy to see that this can only be the case if the role depth of D is greater than the role depth of C_k for all $k \in \mathbb{N}$, or if $D = \bot$. The latter can be ruled out since $\{\mathsf{Homer}, \mathsf{Marge}\} \not\subseteq \bot^i = \emptyset$ and therefore $\bot$ cannot be a model-based most specific concept for $\{\mathsf{Homer}, \mathsf{Marge}\}$. Since any $\mathcal{EL}$ and $\mathcal{EL}^{\bot}$ concept description has finite role depth it is also impossible that D has a role depth that is greater than k for all $k \in \mathbb{N}$. Therefore, such a concept description D cannot exist and $\{\mathsf{Homer}, \mathsf{Marge}\}$ has no model-based most specific concept in i.

The reason that there is no model-based most specific concept in this interpretation is because the interpretation is cyclic. Informally speaking, as soon as there is a cycle in the interpretation we can always find a more specific concept description by increasing role depth, and this process can continue infinitely. One way to deal with cycles in interpretations is to allow cycles in the concept descriptions, too. This brings us to the logics $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$.

We show that model-based most specific concepts exist in both $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$ for all interpretations $i = (\Delta_i, \cdot^i)$ and all sets $X \subseteq \Delta_i$. We distinguish three cases, namely where X is the empty set, where X is a singleton set and where X has at least two elements. For the first case where $X = \emptyset$ the model-based most specific concept in $\mathcal{EL}^{\bot}_{\text{gfp}}$ is obviously

the bottom concept $\bot$. In pure $\mathcal{EL}_{\text{gfp}}$ there is no bottom concept, but there is something similar: There is a concept that encompasses all properties that can be described using $\mathcal{EL}_{\text{gfp}}$. Let $(\mathcal{N}_C, \mathcal{N}_R)$ be a fixed signature. Let $\mathcal{T}_{\text{all}}$ be the TBox

$$\mathcal{T}_{\text{all}} = \{A_{\text{all}} \equiv \prod_{B \in \mathcal{N}_C} B \sqcap \prod_{r \in \mathcal{N}_R} \exists r.A_{\text{all}}\}. \tag{4.2}$$

We define C_{all} to be the $\mathcal{EL}_{\text{gfp}}$-concept $(A_{\text{all}}, \mathcal{T}_{\text{all}})$.

Lemma 4.4. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation for a given signature $(\mathcal{N}_C, \mathcal{N}_R)$. Then C_{all} is the model-based most specific concept of $\emptyset$ in $\mathcal{EL}_{\text{gfp}}$ and $\bot$ is the model-based most specific concept of $\emptyset$ in $\mathcal{EL}^{\bot}_{\text{gfp}}$.*

Proof. To show that C_{all} is the most specific concept for $\emptyset$ we need to prove (M1) and (M2). It trivially holds that $\emptyset \subseteq C^i_{\text{all}}$ and therefore (M1) is true. Assume that $D = (A_D, \mathcal{T}_D)$ is an arbitrary $\mathcal{EL}_{\text{gfp}}$-concept description over the signature $(\mathcal{N}_C, \mathcal{N}_R)$. Let Z be the relation $Z = \{(A, A_{\text{all}}) \mid A \in \mathcal{N}_D(\mathcal{T}_D)\}$. Because of $\text{names}_{\mathcal{T}_{\text{all}}}(A_{\text{all}}) = \mathcal{N}_C$ the relation Z satisfies (S1″). Furthermore, because $A_{\text{all}} \in \text{succ}^r_{\mathcal{T}_{\text{all}}}(A_{\text{all}})$ holds for all $r \in \mathcal{N}_R$ the relation Z also satisfies (S2″). Lastly, Z also satisfies $(A_D, A_{\text{all}}) \in Z$ and therefore (S3″). Thus Z is a simulation from D to C_{all}. Lemma 2.3 implies $C_{\text{all}} \sqsubseteq D$. Therefore, C_{all} is subsumed by every $\mathcal{EL}_{\text{gfp}}$-concept description, in particular this proves (M2) for C_{all}.

The case for $\mathcal{EL}^{\bot}_{\text{gfp}}$ is simpler. First of all, $\emptyset \subseteq \bot^i$ holds. If C is an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description with $\emptyset \subseteq C^i$ then $\bot \sqsubseteq C$ holds trivially. Therefore, $\bot$ is the most specific concept of $\emptyset$. □

In Section 2.4.2 we have seen that $\mathcal{EL}_{\text{gfp}}$-interpretations and cyclic $\mathcal{EL}$-TBoxes share a common structure. In principle, this means that interpretations can be viewed as TBoxes and vice versa. Let $i = (\Delta_i, \cdot^i)$ be an interpretation and G_i its $\mathcal{EL}$-description graph. G_i gives rise to a TBox $\mathcal{T}_{G_i}$ as defined in Definition 2.23. Notice that by definition the defined concept names of $\mathcal{T}_{G_i}$ are the vertices of G_i which are the individuals from Δ_i.[1] Let $x \in \Delta_i$ be an individual. We call the $\mathcal{EL}_{\text{gfp}}$-concept description $C_x = (x, \mathcal{T}_{G_i})$ *the concept description of x in i*, i. e. every individual $x \in \Delta_i$ gives rise to an $\mathcal{EL}_{\text{gfp}}$-concept description. We

[1] This means that we have to violate the convention that concept names are denoted by upper case letters.

show that the concept description of x in i is the model-based most specific concept of $\{x\}$.

Lemma 4.5. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation for a given signature $(\mathcal{N}_C, \mathcal{N}_R)$. Let $x \in \Delta_i$ be an individual, and let $C_x = (x, \mathcal{T}_{G_i})$ be the concept description of x in i. Then C_x is the model-based most specific concept of $\{x\}$ in both $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$.*

Proof. Since i and $\mathcal{T}_{G_i}$ have the same $\mathcal{EL}$-description graph G_i the identity relation id_{Δ_i} on Δ_i is a simulation from C_x to x in i. We can apply Lemma 2.2 to id_{Δ_i} and obtain $\{x\} \subseteq C_x^i$. Therefore C_x satisfies (M1).

Let $D = (A_D, \mathcal{T}_D)$ be another $\mathcal{EL}_{\text{gfp}}$-concept description that satisfies $\{x\} \subseteq D^i$ and let G_D be its $\mathcal{EL}$-description graph. Lemma 2.2 shows that there is a simulation Z from G_D to the description graph G_i such that $(A_D, x) \in Z$. Since G_i is also the description graph of C_x it follows that Z is also a simulation from D to C_x. We can apply Lemma 2.3 to Z and obtain $C_x \sqsubseteq D$. This proves (M2) for $\mathcal{EL}_{\text{gfp}}$.

To prove (M2) for $\mathcal{EL}^{\bot}_{\text{gfp}}$ we distinguish two cases: the case where $D = \bot$ and the case where D is an $\mathcal{EL}_{\text{gfp}}$-concept description. The case $D = \bot$ can be ruled out immediately since $D = \bot$ implies $\{x\} \not\subseteq D^i$. The second case can be treated as for $\mathcal{EL}_{\text{gfp}}$. This shows that C_x is the model-based most specific concept of $\{x\}$ in both $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$. □

Example 4.1. Suppose that we want to compute the model-based most specific concept of $\{\mathsf{Homer}\}$ in the interpretation $i = (\Delta_i, \cdot^i)$ from (4.1). The first step is to translate the $\mathcal{EL}$-description graph of i, which is depicted in Figure 2.2, into a TBox $\mathcal{T}_i$. We obtain

$$\begin{aligned} \mathcal{T}_i = \{&\mathsf{Homer} \equiv \mathsf{Male} \sqcap \exists \mathsf{marriedTo}.\mathsf{Marge}, \\ &\mathsf{Marge} \equiv \mathsf{Female} \sqcap \exists \mathsf{marriedTo}.\mathsf{Homer}\}. \end{aligned} \tag{4.3}$$

Notice that what used to be individual names in Δ_i are now defined concept names in $\mathcal{T}_i$. The $\mathcal{EL}^{\bot}_{\text{gfp}}$-model-based most specific concept for $\{\mathsf{Homer}\}$ is the concept $\{\mathsf{Homer}\}^i = (\mathsf{Homer}, \mathcal{T}_i)$. This concept describes all males that are married to a female that is married to a male, and so on. Analogously, we obtain $\{\mathsf{Marge}\}^i = (\mathsf{Marge}, \mathcal{T}_i)$.

Lemma 4.6. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation for a given signature $(\mathcal{N}_C, \mathcal{N}_R)$ and let $X \subseteq \Delta_i$ be a subset of Δ_i.*

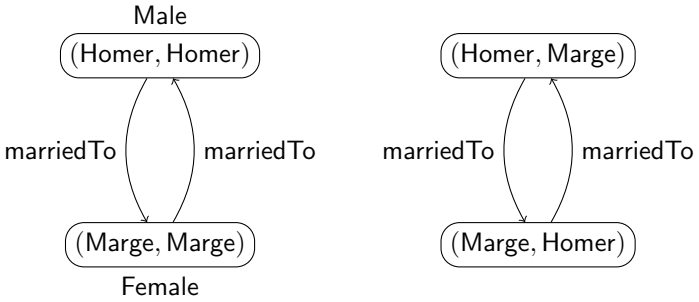

Figure 4.1: $G_i \otimes G_i$

Then the least common subsumer $\mathrm{lcs}\{\{x\}^i \mid x \in X\}$ *exists in both* $\mathcal{EL}_{\mathrm{gfp}}$ *and* $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$*. It is the model-based most specific concept of* X *in both* $\mathcal{EL}_{\mathrm{gfp}}$ *and* $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$*.*

Proof. The existence of least common subsumers has been shown in Lemma 2.4 and Lemma 2.5. The following arguments work for both logics $\mathcal{EL}_{\mathrm{gfp}}$ and $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$, since they only make use of general properties of the model-based most specific concept and the least common subsumer.

By definition $\mathrm{lcs}\{\{x\}^i \mid x \in X\}$ subsumes all concepts $\{x\}^i$ for all $x \in X$. Lemma 4.1 yields $\{x\} \subseteq (\{x\}^i)^i$. Hence all $x \in X$ satisfy $\{x\} \subseteq \left(\mathrm{lcs}\{\{x\}^i \mid x \in X\}\right)^i$ and therefore $X \subseteq \left(\mathrm{lcs}\{\{x\}^i \mid x \in X\}\right)^i$ holds. This proves (M1).

To show (M2) let D be a concept description with $X \subseteq D^i$. Then in particular all $x \in X$ satisfy $\{x\} \subseteq D^i$. Lemma 4.1 yields $\{x\}^i \sqsubseteq D$. Therefore, D is a common subsumer of the concept descriptions $\{x\}^i$ for all $x \in X$. The minimality of the least common subsumer implies $\mathrm{lcs}\{\{x\}^i \mid x \in X\} \sqsubseteq D$ which proves (M2) for $\mathrm{lcs}\{\{x\}^i \mid x \in X\}$. Hence, $\mathrm{lcs}\{\{x\}^i \mid x \in X\}$ is the model-based most specific concept for X. □

Example 4.2. We compute the model-based most specific concept of $\{\mathsf{Homer}, \mathsf{Marge}\}$ in the model $i = (\Delta_i, \cdot^i)$ from (4.1). We already know that $\{\mathsf{Homer}\}^i = (\mathsf{Homer}, \mathcal{T}_i)$ and $\{\mathsf{Marge}\}^i = (\mathsf{Marge}, \mathcal{T}_i)$ with $\mathcal{T}_i$ from (4.3). The model-based most specific concept of $\{\mathsf{Homer}, \mathsf{Marge}\}$ is the

least common subsumer of $\{\mathsf{Homer}\}^i$ and $\{\mathsf{Marge}\}^i$. By Lemma 2.4 the least common subsumer is the product $\{\mathsf{Homer}\}^i \otimes \{\mathsf{Marge}\}^i$. It is obtained from the product of the corresponding description graphs $G_i \otimes G_i$ where G_i is the description graph of $\mathcal{T}_i$ which is depicted in Figure 2.2. Figure 4.1 shows $G_i \otimes G_i$. The corresponding TBox $\mathcal{T}_D$ is

$$\begin{aligned}\mathcal{T}_D = \{&(\mathsf{Homer}, \mathsf{Homer}) = \mathsf{Male} \sqcap \exists \mathsf{marriedTo}.(\mathsf{Marge}, \mathsf{Marge}),\\ &(\mathsf{Marge}, \mathsf{Marge}) = \mathsf{Female} \sqcap \exists \mathsf{marriedTo}.(\mathsf{Homer}, \mathsf{Homer}),\\ &(\mathsf{Marge}, \mathsf{Homer}) = \exists \mathsf{marriedTo}.(\mathsf{Homer}, \mathsf{Marge}),\\ &(\mathsf{Homer}, \mathsf{Marge}) = \exists \mathsf{marriedTo}.(\mathsf{Marge}, \mathsf{Homer})\}.\end{aligned}$$

The defined concept names of $\mathcal{T}_D$ are pairs of individual names from Δ_i. We obtain the concept $((\mathsf{Homer}, \mathsf{Marge}), \mathcal{T}_D)$ as the least common subsumer of $\{\mathsf{Homer}\}^i$ and $\{\mathsf{Marge}\}^i$. This concept is the model-based most specific concept of $\{\mathsf{Homer}, \mathsf{Marge}\}$:

$$\{\mathsf{Homer}, \mathsf{Marge}\}^i = ((\mathsf{Homer}, \mathsf{Marge}), \mathcal{T}_D).$$

It is possible to show that $((\mathsf{Homer}, \mathsf{Marge}), \mathcal{T}_D)$ is equivalent to the concept description $(A, \{A \equiv \exists \mathsf{marriedTo}.A)$. This concept describes all individuals that are followed by an infinite chain of $\mathsf{marriedTo}$ successors.

Together Lemma 4.4, Lemma 4.5 and Lemma 4.6 show that model-based most specific concepts exist in both $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$ for all interpretations and for all sets of individuals. They also provide an effective method to compute the model-based most specific concept.

Theorem 4.7. *Model-based most specific concepts exist both in $\mathcal{EL}_{\text{gfp}}$ and in $\mathcal{EL}^{\bot}_{\text{gfp}}$ for all interpretations $i = (\Delta_i, i)$ and all sets $X \subseteq \Delta_i$. They can be computed effectively.*

To sum up we have shown that model-based most specific concept do not necessarily exist in the logics $\mathcal{EL}$ and $\mathcal{EL}^{\bot}$, mainly because there may be cycles in the model that cannot be expressed by a concept description. This problem can be overcome by allowing greatest-fixpoint semantics. We have shown that for $\mathcal{EL}_{\text{gfp}}$ and $\mathcal{EL}^{\bot}_{\text{gfp}}$ the model-based most specific concept always exists. Results about model-based most specific concept in logics that allow for value restrictions such as $\mathcal{FL}_0$, $\mathcal{FLE}$ and $\mathcal{ALC}$ can be found in [Dis09]. The problems with cyclic models exist in these logics as well, but can be overcome using extensions of the standard logics.

4.1.3 Classical FCA from a DL Perspective

We have already shown that there are parallels between the derivation operators from FCA and interpretation functions and model-based most specific concepts in DL. In this section we shall see that FCA can even be viewed as a syntactic variant of the simple DL that only allows for conjunction and the top concept.[2]

Let $\mathcal{L}_\sqcap$ be the Description Logic that allows only for the concept constructors $\sqcap$ and $\top$ with the usual semantics (cf. Table 2.1). This DL does not use roles at all. Therefore an $\mathcal{L}_\sqcap$-interpretation $i = (\Delta_i, \cdot^i)$ is simply a pair of a domain Δ_i and an interpretation function $\cdot^i : \mathcal{N}_C \to 2^{\Delta_i}$ for a set of concept names $\mathcal{N}_C$.

Let $\mathbb{K} = (G, M, I)$ be a formal context. The set of attributes M can be viewed as a set of concept names, i. e. $M = \mathcal{N}_C$,[3] while the set of objects can be viewed as the domain of an interpretation. Thus $\mathbb{K}$ gives rise to an interpretation $i_\mathbb{K} = (G, \cdot^{i_\mathbb{K}})$ where

$$m^{i_\mathbb{K}} = \{m\}'$$

for every $m \in \mathcal{N}_C$. $\mathbb{K}$ and $i_\mathbb{K}$ are two notationally different representations of the same data. Intuitively, when we take the derivation operator of a set of attributes B then we are looking for objects that have *all* the attributes from B. Therefore, when we view attributes as concept names, it is only natural to associate B with the conjunction $\bigsqcap B$ over all elements of B. Let $B \subseteq M$ be a set of attributes. We define

$$\bigsqcap B = \begin{cases} \bigsqcap_{m \in B} m & \text{if } B \neq \emptyset \\ \top & \text{if } B = \emptyset \end{cases} .$$

The following lemma shows that there is a one to one correspondence between the derivation operators in the context $\mathbb{K}$ and the interpretation function or the model-based most specific concept in the interpretation $i_\mathbb{K}$. No information is lost in the translation.

Lemma 4.8. *Let $\mathbb{K}$ be a context, $A \subseteq G$ a set of objects and $B \subseteq M$ a set of attributes. Then*

$$B' = (\bigsqcap B)^{i_\mathbb{K}}, \tag{4.4}$$

[2] Most authors would not call this logic a DL, since they require DLs to allow for at least one kind of quantifier. We do not adopt this strict notion of DL.

[3] Since attributes are usually denoted by lower case letters, we ignore the convention that concept names should be upper case letters here.

and

$$\bigsqcap A' \equiv A^{i_{\mathbb{K}}}. \tag{4.5}$$

Proof. By definition of the FCA derivation operators it holds that

$$\begin{aligned} B' &= \{g \in G \mid \forall m \in B : gIm\} \\ &= \bigcap_{m \in B} \{g \in G \mid gIm\} = \bigcap_{m \in B} \{m\}' \end{aligned}$$

and by definition of $i_{\mathbb{K}}$

$$B' = \bigcap_{m \in B} m^{i_{\mathbb{K}}} = (\bigsqcap B)^{i_{\mathbb{K}}}.$$

To prove the second statement we need to show that $\bigsqcap A'$ is the model-based most specific concept for A in $i_{\mathbb{K}}$. Lemma 3.1 states that $A \subseteq A''$ and (4.4) shows that $A'' = (\bigsqcap A')^{i_{\mathbb{K}}}$. Thus $A \subseteq (\bigsqcap A')^{i_{\mathbb{K}}}$ holds. This proves (M1) for $\bigsqcap A'$.

To show (M2) let D be an $\mathcal{L}_{\sqcap}$-concept description satisfying $A \subseteq D^{i_{\mathbb{K}}}$. Because $\mathcal{L}_{\sqcap}$ only allows for disjunction D must be of the form $D = \bigsqcap S$ for some set of concept names $S \subseteq M$. From (4.4) we obtain $D^{i_{\mathbb{K}}} = S'$ and therefore $A \subseteq S'$ holds. Lemma 3.1 yields $S \subseteq A'$. It follows that

$$D = \bigsqcap S \sqsupseteq \bigsqcap A'$$

holds. This proves (M2) for $\bigsqcap A'$. Thus $\bigsqcap A'$ is the model-based most specific concept for A in $i_{\mathbb{K}}$, i. e. $\bigsqcap A' \equiv A^{i_{\mathbb{K}}}$. □

We have seen that there is a simple translation from a context to an interpretation. We have shown that the derivation operators can be regarded as a special case of the interpretation function and the model-based most specific concept for the very inexpressive logic $\mathcal{L}_{\sqcap}$. This is of course only a part of the story, since there is much more to FCA than just the derivation operators.

4.2 Induced Contexts

In Section 4.1.3 we have looked at FCA from a DL perspective. In this section we do the opposite and try to incorporate DL concept descriptions in an FCA framework. This approach is not new. In fact, what we

call *induced contexts* in this work has been used under different names in various older works [BGSS07, Rud04, Rud06, Pre00]. This section consists of two parts. In the first part we introduce the main definitions related to induced contexts. In the second part we present a number of technical results that establish a relationship between derivation operators in induced contexts and interpretations and model-based most specific concept in the DL world. The technical part can be skipped by those readers who are only interested in the results and not in the technical details of this thesis.

Main definitions

An induced context is a context whose set of objects is the domain of a finite DL interpretation and whose attributes are concept descriptions. In such a context an object x has the attribute C if x is in the extension of the concept C in the interpretation i. We use $\mathcal{EL}^{\bot}_{\text{gfp}}$ for the formal definitions but they can be defined analogously for other Description Logics.

Definition 4.2 (Induced Context)**.** Let i be a finite $\mathcal{EL}^{\bot}_{\text{gfp}}$-interpretation and M a finite set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. The *context induced by M and i* is the formal context $\mathbb{K} = (G, M, I)$, where $G = \Delta_i$ and

$$I = \{(x, C) \mid C \in M \text{ and } x \in C^i\}.$$

We have already motivated in Section 4.1.3 that a set of attributes $U \subseteq M$ corresponds to the $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description which is obtained as the conjunction over all elements of U. In the other direction we can approximate an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description C by the set of all attributes $D \in M$ that subsume C.

Definition 4.3. Let $\mathbb{K}$ be the context induced by M and i, let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and $U \subseteq M$ a set of attributes. We define

$$\bigsqcap U = \begin{cases} \bigsqcap_{D \in U} D & \text{if } U \neq \emptyset \\ \top & \text{if } U = \emptyset \end{cases}$$

and call this the *concept defined by U*. Conversely, we define

$$\text{pr}_M(C) = \{D \in M \mid C \sqsubseteq D\},$$

and call this the *projection of C to M*.

Table 4.1: The Induced Context $\mathbb{K}$ from Example 4.3

	$\bot$	Female	Male	$\exists$marriedTo.$\top$
Homer			×	×
Marge		×		×

Example 4.3. We consider the model $i = (\Delta_i, \cdot^i)$ from (4.1) again (cf. Figure 2.2). Suppose that we use the set $M = \{\bot, \mathsf{Female}, \mathsf{Male}, \exists\mathsf{marriedTo}.\top\}$ as the set of attributes. The context $\mathbb{K}$ induced by M and i is shown in Table 4.1. The concept description

$$C = \mathsf{Female} \sqcap \exists\mathsf{marriedTo}.\mathsf{Male}$$

can be projected to M. We obtain

$$\mathrm{pr}_M(C) = \{\mathsf{Female}, \exists\mathsf{marriedTo}.\top\}.$$

The concept defined by $\mathrm{pr}_M(C)$ is

$$\bigsqcap \mathrm{pr}_M(C) = \mathsf{Female} \sqcap \exists\mathsf{marriedTo}.\top.$$

Notice that $\mathsf{Female} \sqcap \exists\mathsf{marriedTo}.\top$ is strictly less specific than $\mathsf{Female} \sqcap \exists\mathsf{marriedTo}.\mathsf{Male}$.

While there are infinitely many $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept descriptions the set M contains only a finite number of them. The restriction to a finite set of attributes means that information is lost in the translation. Therefore the projection $\mathrm{pr}_{\mathbb{K}}(C)$ is really just an approximation of C. This means that in the general case it holds that $\bigsqcap \mathrm{pr}_M(C) \not\equiv C$ (cf. Example 4.3). However, since all elements of $\mathrm{pr}_M(C)$ subsume C their conjunction must also subsume C.

Lemma 4.9. *Let $\mathbb{K}$ be the context induced by M and i and let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description. It holds that*

$$C \sqsubseteq \bigsqcap \mathrm{pr}_M(C).$$

Another easy consequence of Definition 4.3 is that the mappings $C \mapsto \mathrm{pr}_M(C)$ and $U \mapsto \bigsqcap U$ are antitone, i. e.

- $C \sqsubseteq D$ implies $\mathrm{pr}_M(D) \subseteq \mathrm{pr}_M(C)$, and
- $U \subseteq V$ implies $\bigsqcap V \sqsubseteq \bigsqcap U$.

Technical Results

In general, not all $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions C can be expressed as the conjunction over a set of attributes from M. Information is lost when we project C to M. In the other direction, where we take the conjunction over a set of attributes $U \subseteq M$ no information is lost. This is illustrated by the following lemma, which looks at the extensions of C and $\bigsqcap U$.

Lemma 4.10. *Let $\mathbb{K} = (\Delta_i, M, I)$ be the context induced by M and i. Every $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description C satisfies*

$$C^i \subseteq \big(\mathrm{pr}_M(C)\big)'.$$

Let $U \subseteq M$ be a set of attributes. Then

$$(\bigsqcap U)^i = U'.$$

Proof. Let $x \in C^i$ be an individual. Since all concept descriptions $D \in \mathrm{pr}_M(C)$ subsume C it holds that $x \in D^i$ for all $D \in \mathrm{pr}_M(C)$. By Definition 4.2 this implies that x has all attributes from $\mathrm{pr}_M(C)$ in the induced context $\mathbb{K}$. Thus $x \in \big(\mathrm{pr}_M(C)\big)'$ holds. This proves $C^i \subseteq \big(\mathrm{pr}_M(C)\big)'$.

To prove $(\bigsqcap U)^i = U'$ let $y \in \Delta_i$ be an individual. We obtain

$$\begin{aligned} y \in U' &\Leftrightarrow \forall D \in U \colon yID \Leftrightarrow \forall D \in U \colon y \in D^i \\ &\Leftrightarrow y \in \bigcap_{D \in U} D^i = (\bigsqcap U)^i. \end{aligned}$$

This proves $(\bigsqcap U)^i = U'$. □

The subsumption $\left(\mathrm{pr}_M(C)\right)' \subseteq C^i$ is not true in the general case (cf. Example 4.3). It is true, however, in the case where C can be expressed in terms of M. We first define formally, what we mean when we say that C can be expressed in terms of M.

Definition 4.4. Let C be a concept description and M a set of $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept descriptions. We say that C *can be expressed in terms of* M if there is a subset $U \subseteq M$ such that $C \equiv \bigsqcap U$.

It is an easy consequence of Definition 4.4 that $C \equiv \bigsqcap \mathrm{pr}_M(C)$ holds if C can be expressed in terms of M. Furthermore we obtain a special case of Lemma 4.10 for a concept description C that can be expressed in terms of M.

Lemma 4.11. *Let $i = (\Delta_i, \cdot^i)$ be a interpretation and M a set of $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept descriptions. Let $\mathbb{K}$ be the context induced by M and i. Every $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept description C that can be expressed in terms of M satisfies*

$$C^i = \left(\mathrm{pr}_M(C)\right)'.$$

Proof. We have already proven $C^i \subseteq \left(\mathrm{pr}_M(C)\right)'$ in Lemma 4.10. Let $x \in \Delta_i$ be an individual. Since C is expressible in terms of M it holds that $C \equiv \bigsqcap \mathrm{pr}_M(C)$. We obtain

$$\begin{aligned} x \in C^i &\Leftrightarrow x \in \left(\bigsqcap \mathrm{pr}_M(C)\right)^i \\ &\Leftrightarrow \forall D \in \mathrm{pr}_M(C) : x \in D^i \\ &\Leftrightarrow \forall D \in \mathrm{pr}_M(C) : xID \\ &\Leftrightarrow x \in \left(\mathrm{pr}_M(C)\right)'. \end{aligned}$$

We have thus shown $C^i = \left(\mathrm{pr}_M(C)\right)'$. □

The previous results only used interpretations on the DL side. We will now start to compare the derivation operator for sets of objects to model-based most specific concepts. We have stated earlier that the results and definitions can also be obtained for other DLs than $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$. The two following results are an exception, as they require a logic where model-based most specific concept are guaranteed to exist.

Lemma 4.12. *Let $i = (\Delta_i, \cdot^i)$ be a interpretation and M a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. Let $\mathbb{K}$ be the context induced by M and i. Every set of individuals $O \subseteq \Delta_i$ satisfies*

$$\text{pr}_M(O^i) = O'.$$

Proof. Let $D \in M$ be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description from M. It holds that

$$\begin{aligned} D \in O' &\Leftrightarrow \forall x \in O : xID \\ &\Leftrightarrow \forall x \in O : x \in D^i \\ &\Leftrightarrow O \subseteq D^i. \end{aligned}$$

Lemma 4.1 (7) yields

$$\begin{aligned} D \in O' &\Leftrightarrow O^i \sqsubseteq D \\ &\Leftrightarrow D \in \text{pr}_M(O^i). \end{aligned}$$

This proves $\text{pr}_M(O^i) = O'$. □

Corollary 4.13. *Let $i = (\Delta_i, \cdot^i)$ be a interpretation and M a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. Let $\mathbb{K}$ be the context induced by M and i. Every subset $U \subseteq M$ satisfies*

$$\text{pr}_M((\bigsqcap U)^{ii}) = U''.$$

Proof. From Lemma 4.10 we obtain

$$(\bigsqcap U)^i = U'$$

and hence

$$\text{pr}_M(((\bigsqcap U)^i)^i) = \text{pr}_M((U')^i). \tag{4.6}$$

Lemma 4.12 implies

$$\text{pr}_M((U')^i) = (U')'. \tag{4.7}$$

Thus, $\text{pr}_M((\bigsqcap U)^{ii}) = U''$ follows from (4.6) and (4.7). □

Induced contexts are an easy way to use DL concept descriptions in an FCA setting. They occur in various previous works in different forms. However, as Example 4.3 shows, information is lost in the transformation since the attributes are only a small subset of the infinitely large set of all concept descriptions. This shows that it is important to select a "good" set of concept descriptions as the attribute set. There are different approaches to this selection. Rudolph's approach [Rud04, Rud06, RVH07] selects all concept descriptions up to a certain role depth. Baader et al. [BGSS07, Ser07] leave the choice to the user who is asked to find concept descriptions that she deems interesting. Our approach to finding interesting concept descriptions is presented in Section 5.2.1.

5 Axiomatization of Finite Models

The Duquenne-Guigues base of a formal context condenses the implicational theory of a formal context into a set of implications of minimal cardinality. In this section we transfer this idea of an implicational base to the DL world and define the notion of a base for an $\mathcal{EL}^{\perp}_{\text{gfp}}$-model. Since there are infinitely many $\mathcal{EL}^{\perp}_{\text{gfp}}$-concept descriptions it is not trivial to prove that a finite base exists for all models, let alone to compute it. This section is divided into three parts. In the first part we prove that a finite base for the $\mathcal{EL}^{\perp}_{\text{gfp}}$-GCIs in a given finite model always exists. In the second part we show that the size of this base can be reduced further by applying results from FCA. As a last step we demonstrate how every finite $\mathcal{EL}^{\perp}_{\text{gfp}}$-base can be transformed into a finite $\mathcal{EL}^{\perp}$-base. The main results from this section have been published in [BD08, BD09].

5.1 Existence of Finite Bases in $\mathcal{EL}^{\perp}_{\text{gfp}}$

Our aim is to extract conceptual knowledge from data – data that is represented in a *closed-world* manner, e. g. given in the form of an $\mathcal{EL}^{\perp}_{\text{gfp}}$-model. Let $(\mathcal{N}_C, \mathcal{N}_R)$ be a signature and $i = (\Delta_i, \cdot^i)$ a model for this signature.[1] In the setting used in this section we assume that the model i is known entirely. We try to find a TBox that describes the complete conceptual knowledge about i. To this purpose we define the notion of a base for the GCIs holding in i.

Definition 5.1 (Base)**.** Let $\mathcal{L}$ be a DL-language. Let $i = (\Delta_i, \cdot^i)$ be a finite interpretation. Let $\mathcal{B}$ be a set of GCIs over the signature $(\mathcal{N}_C, \mathcal{N}_R)$.

[1] Strictly speaking, i is initially only an interpretation and not a model. The goal of this chapter is to obtain a TBox $\mathcal{T}$ that completely describes i and in particular has i as its model. This justifies the small abuse of terminology.

We say that $\mathcal{B}$ is *a base for the $\mathcal{L}$-GCIs holding in i* if it satisfies the following properties.

- $\mathcal{B}$ is *sound* for i, i. e. every GCI from $\mathcal{B}$ holds in i, and
- $\mathcal{B}$ is *complete* for i, i. e. every GCI $C \sqsubseteq D$ that holds in i follows from $\mathcal{B}$.

This notion of completeness is stronger than the one used in [BGSS07], since it takes arbitrary GCIs and not only GCIs of a special type into consideration. By definition, a general TBox is a finite set of GCIs. Therefore, if $\mathcal{B}$ is a finite base for the GCIs holding in i then $\mathcal{B}$ is also a general TBox and i is a model of $\mathcal{B}$. We say that we are *axiomatizing* the model i when we are computing a finite base for it. For now, we allow infinite bases.

One infinite base can be obtained from Lemma 4.3 which is the DL counterpart of Lemma 3.4 from FCA. Here, model-based most specific concepts prove to be fruitful for axiomatizing models for the first time. Let $C \sqsubseteq D$ be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCI that holds in i. Lemma 4.3 shows that $C \sqsubseteq D$ follows from $\{C \sqsubseteq C^{ii}\}$. It is an immediate consequence of Lemma 4.3 and Theorem 4.7.

Corollary 5.1. *Let $i = (\Delta_i, \cdot^i)$ be a model for the signature $(\mathcal{N}_C, \mathcal{N}_R)$. The set of GCIs*

$$\mathcal{B}_0 = \{C \sqsubseteq C^{ii} \mid C \text{ is an } \mathcal{EL}^{\bot}_{\text{gfp}}\text{-concept description over the signature } (\mathcal{N}_C, \mathcal{N}_R), C \neq \bot\}$$

is a base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in i.

We do not need to include $\bot$ as a left-hand side in $\mathcal{B}_0$ since every GCI of the form $\bot \sqsubseteq C$, where C is an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description, holds trivially in any interpretation.

If $\mathcal{N}_R \neq \emptyset$ then there are infinitely many $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions over the signature $(\mathcal{N}_C, \mathcal{N}_R)$. Therefore $\mathcal{B}_0$ is not finite in the general case. The fact that there may be infinitely many $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions makes the search for a finite base for an $\mathcal{EL}^{\bot}_{\text{gfp}}$-model more difficult than the search for a finite implicational base for a formal context. Our first challenge in this section is to show that a finite base for a given

model i does indeed exist and to provide an effective method to compute it. This is done in Section 5.1.2. The proofs in Section 5.1.2 rely on a result from Section 5.1.1 which shows that $\mathcal{B}_0$ is still a base if we remove all GCIs with cyclic left-hand sides.

5.1.1 A Base in $\mathcal{EL}^{\bot}_{\text{gfp}}$ with Only Acyclic Left-Hand Sides

We have seen that using $\mathcal{EL}^{\bot}_{\text{gfp}}$ instead of $\mathcal{EL}^{\bot}$ guarantees the existence of model-based most specific concepts. This allows us to transfer certain ideas such as the derivation operators from FCA to DL. Allowing cyclic concept descriptions on the other hand has drawbacks both of theoretical and of practical nature. From a technical perspective, one of $\mathcal{EL}^{\bot}$'s advantages over $\mathcal{EL}^{\bot}_{\text{gfp}}$ is that its concept descriptions are defined inductively. This allows us to use induction over the structure of a concept description in our proves. The cyclic nature of $\mathcal{EL}^{\bot}_{\text{gfp}}$ on the other hand prevents this. From a practical point of view, the cyclic concept descriptions' semantics is difficult to grasp even for logicians, let alone a domain expert without a degree in logics. An ontology completion formalism that relies on cyclic concept descriptions alone is thus likely to deter potential users.

In this section we examine connections between cyclic and acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. We introduce the notion of an unravelling of a cyclic concept description. Using unravellings we shall be able to prove that for a given model i there is always a base of GCIs whose left-hand sides are acyclic. This result is relevant for the technical parts of Section 5.1.2 (it allows us to use structural induction, etc.). From a practical point of view it is unsatisfying because the final base may still contain cyclic descriptions on the right-hand sides, which is not user friendly. This is addressed in Section 5.3 where we prove the stronger result that a base containing GCIs with cyclic concept descriptions can be converted into a base that uses acyclic concept descriptions exclusively. This conversion also relies on the results about unravellings which are presented in this chapter.

We have already seen that $\mathcal{EL}^{\bot}_{\text{gfp}}$ is more expressive than plain $\mathcal{EL}^{\bot}$. This means that in general when C is an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description we cannot find an equivalent $\mathcal{EL}^{\bot}$-concept description nor can we find an

equivalent acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description. We can only try to find a "good" acyclic approximation for C. What we consider a "good" approximation depends on the model i that we want to axiomatize. An acyclic concept description D is a "good" approximation of C if it has the same interpretation as C in i. The following lemma shows why this level of approximation suffices for our purposes.

Lemma 5.2. *Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description, let $i = (\Delta_i, \cdot^i)$ be a model and let D be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description such that*

- $C \sqsubseteq D$, *and*
- $C^i = D^i$.

Then $C \sqsubseteq C^{ii}$ follows from $\{D \sqsubseteq D^{ii}\}$.

Proof. Let $j = (\Delta_j, \cdot^j)$ be a model in which $D \sqsubseteq D^{ii}$ holds. $C \sqsubseteq D$ implies

$$C^j \subseteq D^j. \tag{5.1}$$

Since $D \sqsubseteq D^{ii}$ holds in j it follows

$$D^j \subseteq (D^{ii})^j. \tag{5.2}$$

$C^{ii} \equiv D^{ii}$ follows from $C^i = D^i$ and therefore

$$(D^{ii})^j = (C^{ii})^j \tag{5.3}$$

holds. From (5.1), (5.2) and (5.3) we obtain that $C^j \subseteq (C^{ii})^j$. Since j is an arbitrary model for which $D \sqsubseteq D^{ii}$ holds, this proves that $C \sqsubseteq C^{ii}$ follows from $\{D \sqsubseteq D^{ii}\}$. □

Lemma 5.2 shows that in order to obtain a base that only uses acyclic left-hand sides, it suffices to show that every $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description has a good acyclic approximation. We use unravellings to find these approximations. An unravelling is essentially the transformation of a cyclic, finite, directed graph into an acyclic but potentially infinite tree. Unravellings are defined for $\mathcal{EL}$-description graphs. Let $G = (V, E, L)$ be an $\mathcal{EL}$-description graph. Every directed path in G between two nodes $A \in V$ and $B \in V$ can be described by a word $\mathbf{w} = A_0 r_0 A_1 r_1 \dots r_{n-1} A_n$ where $A_0 = A$, $A_n = B$, $A_k \in V$, $r_k \in \mathcal{N}_R$, and the tuple (A_k, r_k, A_{k+1})

is an edge $(A_k, r_k, A_{k+1}) \in E$ for all $k \in \{0, \ldots, n-1\}$. A path is allowed to loop, i. e. it can use a vertex or even an edge several times. We call n the *length of the path* $\mathbf{w}$. We also allow paths of length 0, which are words of the form $\mathbf{w} = A$ where $A \in \mathsf{N}_C$ is a concept name. We call the last vertex on the path $\mathbf{w}$ the *destination* of $\mathbf{w}$ and denote it by $\delta(\mathbf{w})$, i. e.

$$\delta(\mathbf{w}) = \begin{cases} B, & \text{if } \mathbf{w} = \mathbf{w}' r B \text{ for some path } \mathbf{w}' \text{ in } G, r \in \mathsf{N}_R, B \in \mathsf{N}_C \\ A, & \text{if } \mathbf{w} = A. \end{cases}$$

Definition 5.2 (Unravelling)**.** Let G be a finite $\mathcal{EL}$-description graph. The *unravelling of* $G = (V, E, L)$ *at a vertex* $A \in V$ is the $\mathcal{EL}$-description graph $G^\infty = (V^\infty, E^\infty, L^\infty)$ where

- V^∞ is the set of all directed paths in G that start in A, and
- $E^\infty = \{(\mathbf{w}, r, \mathbf{w}rB) \mid \mathbf{w}, \mathbf{w}rB \in V^\infty\}$, i. e. two paths are connected by an r-edge in G^∞ if one is obtained from the other by adding an r-edge in G, and
- $L^\infty = L(\delta(\mathbf{w}))$.

The *unravelling of* G *up to depth* d *at* A is the $\mathcal{EL}$-description graph G^d that is obtained from G^∞ by removing all vertices whose path length is greater than d and the corresponding edges. That means we prune all branches of G^∞ whose length exceeds d. Formally, we define $G^d = (V^d, E^d, L^d)$, where

- $V^d = \{\mathbf{w} \in V^\infty \mid \text{length}(\mathbf{w}) \leq d\}$, and
- $E^d = E^\infty \cap (V^d \times \mathsf{N}_R \times V^d)$, and
- $L^d = L^\infty|_{V^d}$ is the restriction of L^∞ to the vertices in V^d.

TBoxes and finite $\mathcal{EL}$-description graphs share a common structure and one can be converted to the other using Definition 2.23. Because of this common structure unravellings can also be defined for TBoxes and $\mathcal{EL}_{\text{gfp}}$-concept descriptions.

Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, $G_\mathcal{T}$ its description graph, $A \in \mathsf{N}_D(\mathcal{T})$ a defined concept and $d \in \mathbb{N}$ a natural number. Let $G^d_\mathcal{T}$ be the unravelling of $G_\mathcal{T}$ up to depth d at A. The unravelling of a finite $\mathcal{EL}$-description graph

up to a finite role depth d is always finite. Therefore the unravelled graph $G_{\mathcal{T}}^d$ can be translated to an general $\mathcal{EL}$-TBox $\mathcal{T}^d$.[2] We speak of the *unravelling of* $\mathcal{T}$ up to role depth d at the defined concept A. Let $C = (A, \mathcal{T})$ be an $\mathcal{EL}_{\text{gfp}}$-concept description. By C_d we denote the concept description $C_d = (A, \mathcal{T}^d)$, where $\mathcal{T}^d$ is the unravelling of $\mathcal{T}$ up to role depth d at A. We call C_d the *unravelling of* C *up to role depth* d. The resulting description is always acyclic and therefore equivalent to an $\mathcal{EL}$-concept description.

Unravellings can be defined for $\mathcal{EL}_{\text{gfp}}$-concept descriptions and hence also for all $\mathcal{EL}_{\text{gfp}}^{\bot}$-concept descriptions except the bottom concept $\bot$. For matters of convenience we define the unravelling of $\bot$ up to any role depth d to be $\bot$.

Example 5.1. Let the $\mathcal{EL}_{\text{gfp}}$-concept description $\mathsf{MarriedFather} = (A, \mathcal{T})$ be defined by

$$\begin{aligned}\mathcal{T} = \{A &\equiv \mathsf{Male} \sqcap \exists \mathsf{hasChild}.C \sqcap \exists \mathsf{marriedTo}.B,\\ B &\equiv \exists \mathsf{hasChild}.C \sqcap \exists \mathsf{marriedTo}.A,\\ C &\equiv \top\}.\end{aligned}$$

The description graph of $\mathsf{MarriedFather}$ is depicted in Figure 5.1. There are infinitely many paths leading from A to any other node. This is because a path can traverse the loop between A and B an arbitrary number of times before reaching its destination. Therefore we obtain an infinitely large unravelling (Figure 5.3).

Figure 5.2 depicts the description graph of $\mathsf{MarriedFather}^1$ which is the unravelling of $\mathsf{MarriedFather}$ up to role depth 1. Two things can be seen from the figure. First, $\mathsf{MarriedFather}^1$ is acyclic and finite. Therefore, it is equivalent to an $\mathcal{EL}$-concept description, namely

$$\mathsf{MarriedFather}^1 \equiv \mathsf{Male} \sqcap \exists \mathsf{hasChild}.\top \sqcap \exists \mathsf{marriedTo}.\top.$$

Second, $\mathsf{MarriedFather}^1$ is less specific than $\mathsf{MarriedFather}$. For example $\mathsf{MarriedFather}$ specifies that the married father's spouse is also married with a child while the concept description $\mathsf{MarriedFather}^1$ does not specify this.

[2] Infinite graphs would give rise to infinite TBoxes which we do not allow in this thesis.

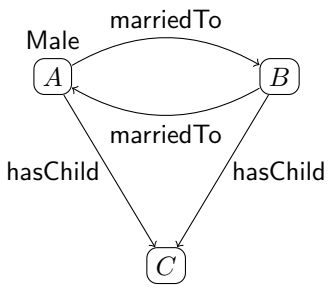

Figure 5.1: $\mathcal{EL}$-Description Graph of MarriedFather

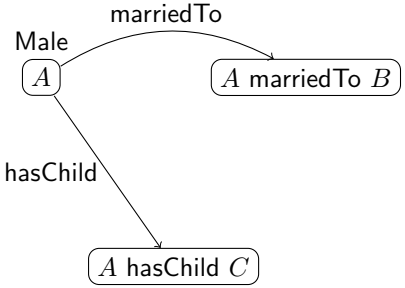

Figure 5.2: Unravelling up to role depth 1

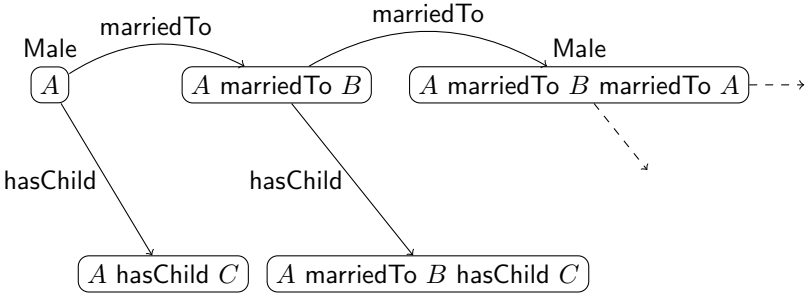

Figure 5.3: Unravelling of MarriedFather at the vertex A

$C \sqsubseteq C_d$ always holds. This follows from Lemma 2.2 and the following lemma.

Lemma 5.3. *Let $G = (V, E, L)$ be an $\mathcal{EL}$-description graph, $A \in V$ a vertex from G, $d \in \mathbb{N}$, and $G^d = (V^d, E^d, L^d)$ the unravelling of G up to role depth d at A. Then*

$$Z^d = \{(\mathbf{w}, \delta(\mathbf{w})) \mid \mathbf{w} \in V^d\}$$

is a simulation from G^d to G.

Proof. To prove that Z^d is a simulation we need to prove (S1) and (S2) from Definition 2.24. *(S1)* Let $(\mathbf{w}, \delta(\mathbf{w})) \in Z^d$ be a pair in the relation Z^d. By Definition 5.2 it holds that $L^d(\mathbf{w}) = L(\delta(\mathbf{w}))$. Hence in particular $L^d(\mathbf{w}) \subseteq L(\delta(\mathbf{w}))$ holds. *(S2)* Let $(\mathbf{w}, \delta(\mathbf{w})) \in Z^d$ be a pair in the relation Z^d and $(\mathbf{w}, r, \mathbf{w}rB) \in E^d$ an edge in G^d. Since $\mathbf{w}rB$ is a vertex in V^d it also must be a path in G. Therefore there must be an r-edge in G that connects the destination of $\mathbf{w}$ to B, i. e. $(\delta(\mathbf{w}), r, B) \in E$. Hence, B satisfies both $(\mathbf{w}rB, B) \in Z^d$ and $(\delta(\mathbf{w}), r, B) \in E$. This proves (S2) for Z^d. Since both (S1) and (S2) hold, Z^d is a simulation from G^d to G. □

Example 5.1 shows that in general $C_d \not\sqsubseteq C$ for cyclic $\mathcal{EL}_{\text{gfp}}$-concept descriptions C. Intuitively, this holds because information from C is lost during the pruning step, thereby making C_d less specific. Obviously, the larger the role depth at which we prune the less information is lost. We are now interested in the question whether we can make d large enough such that C_d and C have the same interpretation in a fixed model i. We first prove a specialized version of Lemma 2.2 that characterizes the instance relation using simulations.

Lemma 5.4. *Let $C = (A_C, \mathcal{T}_C)$ be an acyclic $\mathcal{EL}_{\text{gfp}}$-concept description and $i = (\Delta_i, \cdot^i)$ a finite interpretation. Let $x \in C^i$ be an individual in the interpretation of C. Then there exists a simulation Z from C to x in i with the additional property that for every defined concept $B \in \mathcal{N}_D(\mathcal{T}_C)$ there is exactly one individual y satisfying $(B, y) \in Z$.*

Proof. Remember that acyclic $\mathcal{EL}_{\text{gfp}}$-concept descriptions correspond to $\mathcal{EL}$-concept descriptions and that their $\mathcal{EL}$-description graphs are trees. This means we can use structural induction over the structure of C.

Base Case Assume $C = B$ for some concept name $B \in \mathcal{N}_C$. Remember that in Section 2.4.1 we have introduced B as a shorthand for the $\mathcal{EL}_{\text{gfp}}$-concept description $B = (A_B, \mathcal{T}_B)$, where $T_B = \{A_B \equiv B\}$. Then $Z_B = \{(A_B, x)\}$ is a simulation from $C = B$ to x in i with the property that for every concept name $A \in \mathcal{N}_D(\mathcal{T}_C)$ there is exactly one individual y which satisfies $(A, y) \in Z_B$. Likewise, the simulation $Z_\top = \{(A_\top, x)\}$ can be used if $C = \top = (A_\top, \{A_\top \equiv \top\})$.

Step Case 1 Assume that $C = \exists r.D$ holds for some role name $r \in \mathcal{N}_R$ and some $\mathcal{EL}_{\text{gfp}}$-concept description $D = (A_D, \mathcal{T}_D)$ whose description graph is a tree. Since $x \in C^i$ holds there must be some $y \in \Delta_i$ such that $(x, y) \in r^i$ and $y \in D^i$. By the induction hypothesis there is a simulation Z_D from D to y in i. We leave it to the reader to check that $\{(A_C, x)\} \cup Z_D$ is a simulation from C to x in i with the desired property.

Step Case 2 Assume that $C = D \sqcap E$ for some $\mathcal{EL}_{\text{gfp}}$-concept descriptions $D = (A_D, \mathcal{T}_D)$ and $E = (A_E, \mathcal{T}_E)$, where $\mathcal{N}_D(\mathcal{T}_D)$ and $\mathcal{N}_D(\mathcal{T}_E)$ are disjoint. Since $x \in C^i$ holds, x satisfies $x \in D^i$ and $x \in E^i$. Therefore there are simulations Z_D from D to x in i and Z_E from E to x in i. Again, one can check that $Z_D \cup Z_E \cup \{(A_C, x)\}$ is a simulation from C to x in i with the desired property. □

Lemma 5.5. *Let $C = (A, \mathcal{T})$ be an $\mathcal{EL}_{\text{gfp}}$-concept description and $i = (\Delta_i, \cdot^i)$ a finite interpretation. Let $d = |\mathcal{N}_D(\mathcal{T})| \cdot |\Delta_i| + 1$. Let $C_d = (A, \mathcal{T}^d)$ be the unravelling of C up to role depth d. Then $x \in C^i_d$ implies $x \in C^i$.*

Proof. Let $G_\mathcal{T} = (\mathcal{N}_D(\mathcal{T}), E_\mathcal{T}, \text{names}_\mathcal{T})$, $G^d = (V^d, E^d, L^d)$ and $G_i = (\Delta_i, E_i, \text{names}_i)$ be the $\mathcal{EL}$-description graphs of $\mathcal{T}$, $\mathcal{T}^d$ and i, respectively. The unravelled TBox $\mathcal{T}^d$ has a tree shaped $\mathcal{EL}$-description graph G^d and C_d is acyclic. Lemma 5.4 and $x \in C^i_d$ imply that there is a simulation Z from G^d to G_i which satisfies $(A, x) \in Z$ and for every vertex $\mathbf{w} \in V^d$ there is exactly one individual $y \in \Delta_i$ such that $(\mathbf{w}, y) \in Z$. Z gives rise to a function z that maps every vertex $\mathbf{w} \in V^d$ to $z(\mathbf{w}) = y$ where $(\mathbf{w}, y) \in Z$.

Since G^d is tree shaped with the root A there is exactly one directed path from A to any given vertex $\mathbf{w} \in V^d$. If this path has length d then the pigeonhole principle implies that it will always contain two vertices $\mathbf{w}_1$ and $\mathbf{w}_2$ such that both $z(\mathbf{w}_1) = z(\mathbf{w}_2)$ and $\delta(\mathbf{w}_1) = \delta(\mathbf{w}_2)$,

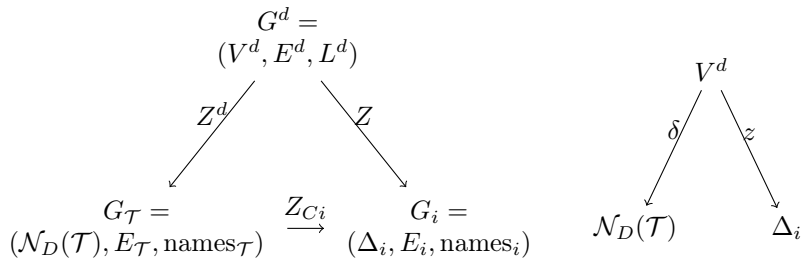

Figure 5.4: Diagram of Simulations and Mappings from the Proof of Lemma 5.5

because there are only $|\mathcal{N}_D(\mathcal{T})| \cdot |\Delta_i| = d-1$ possible values for the pair $(\delta(\mathbf{w}), z(\mathbf{w}))$. We exploit this fact to construct a simulation from $G_\mathcal{T}$ to G_i.

Let $\bar{V} \subseteq V^d$ be the set of all vertices $\mathbf{w} \in V^d$ such that there are no two distinct vertices $\mathbf{w}_1$ and $\mathbf{w}_2$ on the path from A to $\mathbf{w}$ that have the same values $\delta(\mathbf{w}_1) = \delta(\mathbf{w}_2)$ and $z(\mathbf{w}_1) = z(\mathbf{w}_2)$. Because of the above argument $\bar{V}$ contains only vertices whose path distance from A is strictly less than d. If $\mathbf{w} \in \bar{V}$ is a vertex in $\bar{V}$ then every vertex $\mathbf{w}'$ on the path from A to $\mathbf{w}$ in G^d is also an element of $\bar{V}$. Hence $\bar{V}$ spans a subtree of G^d. Define Z_{Ci} to be the relation

$$Z_{Ci} = \{(\delta(\mathbf{w}), z(\mathbf{w})) \mid \mathbf{w} \in \bar{V}\}.$$

Figure 5.4 shows a diagram of the simulations and mappings that occur in this proof. We show that Z_{Ci} is a simulation from $G_\mathcal{T}$ to G_i.

(S1) By definition L^d satisfies $L^d(\mathbf{w}) = \text{names}_\mathcal{T}(\delta(\mathbf{w}))$ for all $\mathbf{w} \in V^d$. Since Z is a simulation it holds that $L^d(\mathbf{w}) \subseteq \text{names}_i(z(\mathbf{w}))$ for all $\mathbf{w} \in V^d$. Thus $\text{names}_\mathcal{T}(\delta(\mathbf{w})) \subseteq \text{names}_i(z(\mathbf{w}))$ holds for all pairs $(\delta(\mathbf{w}), z(\mathbf{w})) \in Z_{Ci}$. This proves (S1) for Z_{Ci}.

(S2) The proof of this property is illustrated in Figure 5.5. Let $(\delta(\mathbf{w}), z(\mathbf{w})) \in Z_{Ci}$ be a pair in Z_{Ci} and let $B \in \mathcal{N}_D(\mathcal{T})$ be a defined concept name that satisfies $(\delta(\mathbf{w}), r, B) \in E_\mathcal{T}$. Since $\mathbf{w}$ is contained in $\bar{V}$ its path distance from A is strictly less than d. Thus the path length of $\mathbf{w}rB$ is at most d. Hence, $\mathbf{w}rB$ is contained in V^d. By definition E^d

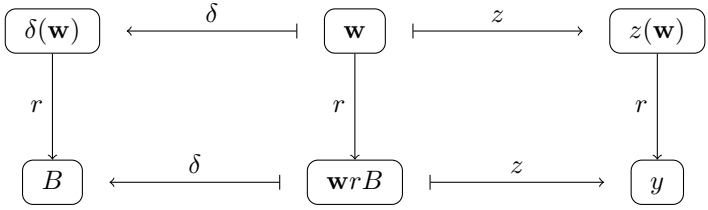

Figure 5.5: Diagram for the Proof of Property (S2) in Lemma 5.5

contains $(\mathbf{w}, \mathbf{w}rB)$. Let $y = z(\mathbf{w}rB)$ be the one individual in Δ_i that satisfies $(\mathbf{w}rB, y) \in Z$. Since (S2) holds for Z the individual y must satisfy $(z(\mathbf{w}), r, y) \in E_i$.

We distinguish the two cases where $\mathbf{w}rB \in \bar{V}$ and where $\mathbf{w}rB \notin \bar{V}$. If $\mathbf{w}rB \in \bar{V}$ holds then y satisfies $(B, y) = (\delta(\mathbf{w}rB), z(\mathbf{w}rB)) \in Z_{Ci}$. Together with $(z(\mathbf{w}), r, y) \in E_i$ this proves (S2) for Z_{Ci}.

If $\mathbf{w}rB \notin \bar{V}$ holds then there must be some vertex $\mathbf{w}' \neq \mathbf{w}rB$ on the path from A to $\mathbf{w}rB$ in G^d such that $z(\mathbf{w}') = z(\mathbf{w}rB)$ and $\delta(\mathbf{w}') = \delta(\mathbf{w}rB)$. Since $\mathbf{w} \in \bar{V}$ holds all vertices on the path from A to $\mathbf{w}$ in G^d are in $\bar{V}$. Hence $\mathbf{w}' \in \bar{V}$ holds. This implies $(\delta(\mathbf{w}'), z(\mathbf{w}')) \in Z_{Ci}$. We obtain $(B, y) = (\delta(\mathbf{w}rB), z(\mathbf{w}rB)) = (\delta(\mathbf{w}'), z(\mathbf{w}')) \in Z_{Ci}$. Together with $(z(\mathbf{w}), r, y) \in E_i$ this proves (S2) for Z_{Ci}.

Since Z_{Ci} satisfies both (S1) and (S2) it is a simulation from $G_{\mathcal{T}}$ to G_i. Z_{Ci} contains the pair $(A, x) = (\delta(A), z(A))$. Therefore Lemma 2.2 proves that $x \in C^i$ holds. □

Example 5.2. The reader may wonder why in the proof of Lemma 5.5 we construct the simulation Z_{Ci} only from the vertices in $\bar{V}$ and not from the full set V, i. e. why the relation

$$Z'_{Ci} = \{(\delta(\mathbf{w}), z(\mathbf{w})) \mid \mathbf{w} \in V\}.$$

is not necessarily a simulation. We provide a simple example to illustrate this. Let $C = (A, \{A \equiv \exists r.A\}$ be an $\mathcal{EL}_{\text{gfp}}$-concept description and let $i = (\{x, y\}, \cdot^i)$ be the model defined by $r^i = \{(x, x), (x, y)\}$ (cf. Figure 5.6). Clearly, x satisfies $x \in C^i$. The unravelling of C up to

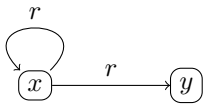

Figure 5.6: The Model from Example 5.2

depth $|\mathcal{N}_D(\mathcal{T}_C)| \cdot |\Delta_i| + 1 = 3$ is

$$\begin{aligned} C_3 = (A, \{&A \equiv \exists r.(ArA), \\ &(ArA) \equiv \exists r.(ArArA), \\ &(ArArA) \equiv \top\}). \end{aligned}$$

One possible simulation from C_3 to x in i is

$$Z = \{(A, x), (ArA, x), (ArArA, y)\}.$$

Since in the unravelling C_3 the defined concept name $(ArArA)$ has no successors the simulation Z can map $(ArArA)$ to y, which also has no successors. However, the relation

$$Z'_{Ci} = \{(A, x), (A, x), (A, y)\}$$

violates (S2), since A has an r-successor in $\mathcal{T}_C$ but y has no r-successors in i. If we use only the vertices form $\bar{V}$ we obtain Z_{Ci}

$$Z_{Ci} = \{(A, x)\}$$

which is a simulation from C to x in i, which is what we expected from Lemma 5.5.

Lemma 5.5 shows that if i is a fixed model and C is an $\mathcal{EL}_{\text{gfp}}$-concept description then there is an acyclic concept description D that approximates C closely enough, such that C and D have the same interpretation in i. It is a corollary that the same holds true for $\mathcal{EL}^{\bot}_{\text{gfp}}$.

Corollary 5.6. *Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and let i be a finite model. Then there is an acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description D such that $C^i = D^i$ and $C \sqsubseteq D$.*

Proof. The case where $C = \bot$ is trivial, since we consider $\bot$ itself to be an acyclic concept description. If $C \neq \bot$ then C is an $\mathcal{EL}_{\text{gfp}}$-concept description $C = (A, \mathcal{T})$. Lemma 5.5 proves that $C^i_d = C^i$ and $C \sqsubseteq C_d$ hold for $d = |\mathcal{N}_D(\mathcal{T})| \cdot |\Delta_i| + 1$. □

The main result of this section follows immediately from Corollary 5.6 and Lemma 5.2. It states that for a given model i we can always find a base whose left-hand sides are acyclic. Since there are infinitely many acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions this is still not a finite base.

Theorem 5.7. *Let $i = (\Delta_i, \cdot^i)$ be a interpretation. The set of GCIs*

$$\mathcal{B}_1 = \{D \sqsubseteq D^{ii} \mid D \text{ is an acyclic } \mathcal{EL}^{\bot}_{\text{gfp}}\text{-concept description}, D \neq \bot\}$$

is a base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in i.

Proof. Soundness of $\mathcal{B}_1$ follows from the fact that $\mathcal{B}_1$ is a subset of $\mathcal{B}_0$. Corollary 5.6 and Lemma 5.2 imply that every GCI in $\mathcal{B}_0$ follows from $\mathcal{B}_1$. Completeness of $\mathcal{B}_1$ follows from the completeness of $\mathcal{B}_0$. □

5.1.2 Finite Bases

In the general case the base $\mathcal{B}_1$ is infinite, because there are infinitely many acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions and thus infinitely many left-hand sides. In this section we present a base $\mathcal{B}_2$ that is finite, but uses both cyclic left-hand sides and right-hand sides. The proofs in this section and the following rely on the proofs from Section 4.2. Readers who have skipped Section 4.2 are advised to read it now if they are interested in the technical details of this section.

The left-hand sides of the base $\mathcal{B}_2$ come from a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions Λ_i. The elements of Λ_i are conjunctions over the elements of a basic set M_i.

Definition 5.3. Let i be a finite $\mathcal{EL}^{\bot}_{\text{gfp}}$-model. The sets M_i and Λ_i are defined as follows

$$M_i = \{\bot\} \cup \mathcal{N}_C \cup \{\exists r.X^i \mid r \in \mathcal{N}_R \text{ and } X \subseteq \Delta_i, X \neq \emptyset\}$$

and

$$\Lambda_i = \{\bigsqcap U \mid U \subseteq M_i\}.$$

Since Δ_i is finite it has only finitely many subsets $X \subseteq \Delta_i$. Hence, both M_i and Λ_i are finite, too. We prove that $\mathcal{B}_0$ remains complete when we remove all GCI whose left-hand sides are not in Λ_i. Since Λ_i is finite the resulting set of implications would also be finite.

The model-based most specific concepts and hence also the elements of M_i are not unique, but only unique up to equivalence. In none of the following proofs in this chapter is it relevant which equivalent representation of M_i we use. Formally, let $\bar{M}$ be a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. We say that M_i subsumes $\bar{M}$ up to equivalence (denoted by $\bar{M} \dot{\subseteq} M_i$) if for each concept description $C \in \bar{M}$ there is an equivalent concept description $D \in M_i$. We say that $\bar{M}$ and M_i are *equal up to equivalence of elements* if $\bar{M} \dot{\subseteq} M_i$ and $M_i \dot{\subseteq} \bar{M}$ hold. All following results also hold if we replace M_i by a set $\bar{M}$ that is equal up to equivalence of elements.

A first technical result is that all model-based most specific concepts are contained in Λ_i. To show this we introduce lower approximations for $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. In Section 4.2 we have introduced projections onto an attribute set M. One way to approximate an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description description C by a description from Λ_i is to take the conjunction over $\text{pr}_{M_i}(C)$. This kind of approximation can be viewed as least upper bounds for C among the concepts from Λ_i. We introduce a second type of approximation, that we call *lower approximation*.[3]

Definition 5.4 (Lower Approximation)**.** Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and $i = (\Delta_i, \cdot^i)$ an interpretation. If C is an $\mathcal{EL}_{\text{gfp}}$-concept description $C = (A_C, \mathcal{T}_C)$ then there is a set of concept names $U \subseteq \mathcal{N}_C$ and a set Π containing pairs of role names and $\mathcal{EL}_{\text{gfp}}$-concept descriptions such that

$$C = \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E.$$

In this case the *lower approximation* of C is defined as

$$\text{approx}_i(C) = \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E^{ii}.$$

For $C = \bot$ the *lower approximation* of C is defined as $\text{approx}_i(C) = \bot$.

[3] One can show that the *lower approximation* of C is the least specific concept description from Λ_i that is subsumed by C, i. e. it is the greatest lower bound for C in Λ_i with respect to $\sqsubseteq$. This is a property that we will not use in this thesis.

Clearly for every concept description C it holds that $\mathrm{approx}_i(C)$ is in Λ_i. This is equivalent to saying that $\mathrm{approx}_i(C)$ can be expressed in terms of M_i.

Lemma 5.8. *Let C be an $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept description and $i = (\Delta_i, \cdot^i)$ an interpretation. It holds that*

$$C^{ii} \sqsubseteq \mathrm{approx}_i(C) \sqsubseteq C$$

Proof. If $C = \bot$ then $C^{ii} = \bot$ and $\mathrm{approx}_i(C) = \bot$. Therefore $C^{ii} \sqsubseteq \mathrm{approx}_i(C) \sqsubseteq C$ holds trivially. We now assume that

$$C = \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E$$

holds for a set of concept names $U \subseteq \mathcal{N}_C$ and a set Π containing pairs of role names and $\mathcal{EL}_{\mathrm{gfp}}$-concept descriptions.

Lemma 4.1 yields $E^{ii} \sqsubseteq E$ for all pairs $(r, E) \in \Pi$ and hence the monotony of existential quantifiers implies $\exists r.E^{ii} \sqsubseteq \exists r.E$ for all pairs $(r, E) \in \Pi$. The monotony of conjunctions implies

$$\begin{aligned} \mathrm{approx}_i(C) &= \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E^{ii} \\ &\sqsubseteq \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E = C. \end{aligned} \tag{5.4}$$

Let us look at the set $C^i \subseteq \Delta_i$. It holds that

$$C^i = \left(\bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E \right)^i .$$

From Lemma 4.2 we obtain

$$\begin{aligned} C^i &= \left(\bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E^{ii} \right)^i \\ &= (\mathrm{approx}_i(C))^i \end{aligned}$$

and thus in particular $C^i \subseteq (\text{approx}_i(C))^i$. Lemma 4.1 (7) shows that

$$C^{ii} \sqsubseteq \text{approx}_i(C). \tag{5.5}$$

The claim follows from (5.4) and (5.5). □

The fact that all concept descriptions of the form X^i, where $X \subseteq \Delta_i$, are in Λ_i is a corollary of the previous lemma.

Lemma 5.9. *For every set $X \subseteq \Delta_i$, the model-based most specific concept X^i in $\mathcal{EL}^{\bot}_{\text{gfp}}$ can be expressed in terms of M_i.*

Proof. From Lemma 5.8 we obtain that

$$(X^i)^{ii} \sqsubseteq \text{approx}_i(X^i) \sqsubseteq X^i.$$

Lemma 4.1 states that $X^{iii} \equiv X^i$ and hence the above inequality collapses to

$$(X^i)^{ii} \equiv \text{approx}_i(X^i) \equiv X^i.$$

Since $\text{approx}_i(X^i)$ can be expressed in terms of M_i the concept description X^i can also be expressed in terms of M_i. □

Theorem 5.10. *Let i be a finite $\mathcal{EL}^{\bot}_{\text{gfp}}$-model. The set*

$$\mathcal{B}_2 = \{C \sqsubseteq C^{ii} \mid C \in \Lambda_i\}$$

is a finite base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in i.

Proof. $\mathcal{B}_2$ is sound for i since it is a subset of $\mathcal{B}_0$ which is sound for i. Furthermore, $\mathcal{B}_2$ is finite since Λ_i is finite. In Section 5.1.1 we have presented the base $\mathcal{B}_1$ which uses only acyclic left-hand sides. Since $\mathcal{B}_1$ is complete it suffices to show that all GCIs from $\mathcal{B}_1$ follow from $\mathcal{B}_2$. The GCIs from $\mathcal{B}_1$ are of the form $D \sqsubseteq D^{ii}$, where D is acyclic and $D \neq \bot$ holds. The acyclic nature of D allows us to use induction over the structure of D.

Base Case 1: If $D = \top$ then $D = \bigsqcap \emptyset$ and thus $D \in \Lambda_i$. Hence, $\mathcal{B}_2$ contains $D \sqsubseteq D^{ii}$. In particular $D \sqsubseteq D^{ii}$ follows from $\mathcal{B}_2$.

Base Case 2: If $D = A$ for some concept name $A \in \mathcal{N}_C$ then $D = \bigsqcap\{A\} \in \Lambda_i$. Therefore, $D \sqsubseteq D^{ii} \in \mathcal{B}_2$ holds and thus $D \sqsubseteq D^{ii}$ follows from $\mathcal{B}_2$.

Step Case 1: Let $D = E \sqcap F$ for some $\mathcal{EL}_{\text{gfp}}$-concept descriptions E and F where by induction hypothesis $E \sqsubseteq E^{ii}$ and $F \sqsubseteq F^{ii}$ follow from $\mathcal{B}_2$. Let $j = (\Delta_j, \cdot^j)$ be a model in which all GCIs from $\mathcal{B}_2$ hold. We obtain

$$D^j = (E \sqcap F)^j = E^j \cap F^j \tag{5.6}$$

Since all GCIs from $\mathcal{B}_2$ hold in j and $E \sqsubseteq E^{ii}$ follows from $\mathcal{B}_2$ it follows that j satisfies $E^j \subseteq (E^{ii})^j$. Likewise j satisfies $F^j \subseteq (F^{ii})^j$. Thus, it holds that

$$E^j \cap F^j \subseteq (E^{ii})^j \cap (F^{ii})^j = (E^{ii} \sqcap F^{ii})^j. \tag{5.7}$$

Lemma 5.9 shows that $E^{ii} \sqcap F^{ii}$ is equivalent to some concept description in Λ_i. Thus $\mathcal{B}_2$ contains $E^{ii} \sqcap F^{ii} \sqsubseteq (E^{ii} \sqcap F^{ii})^{ii}$ or an equivalent GCI. Since all GCIs from $\mathcal{B}_2$ hold in j we obtain

$$(E^{ii} \sqcap F^{ii})^j \subseteq ((E^{ii} \sqcap F^{ii})^{ii})^j. \tag{5.8}$$

Lemma 4.2 proves $(E^{ii} \sqcap F^{ii})^i = (E \sqcap F)^i$ and thus

$$((E^{ii} \sqcap F^{ii})^{ii})^j = ((E \sqcap F)^{ii})^j = (D^{ii})^j. \tag{5.9}$$

(5.6), (5.7), (5.8), and (5.9) show that $D^j \subseteq (D^{ii})^j$ holds, i. e. $D \sqsubseteq D^{ii}$ holds in j. The only assumption we have made about j was that all GCIs from $\mathcal{B}_2$ hold in j. Therefore, $D \sqsubseteq D^{ii}$ follows from $\mathcal{B}_2$.

Step Case 2: Let $D = \exists r.E$ for some $\mathcal{EL}_{\text{gfp}}$-concept description E where by induction hypothesis $E \sqsubseteq E^{ii}$ follows from $\mathcal{B}_2$. Let $j = (\Delta_j, \cdot^j)$ be a model in which all GCIs from $\mathcal{B}_2$ hold. Let x be an individual from Δ_j. Then

$$\begin{aligned} x \in D^j &\Leftrightarrow x \in (\exists r.E)^j \\ &\Leftrightarrow \exists y \in E^j : (x, y) \in r^j \end{aligned} \tag{5.10}$$

Since $E \sqsubseteq E^{ii}$ follows from $\mathcal{B}_2$, j must satisfy $E^j \subseteq (E^{ii})^j$. We obtain

$$\begin{aligned} \exists y \in E^j : (x, y) \in r^j &\Rightarrow \exists y \in (E^{ii})^j : (x, y) \in r^j \\ &\Leftrightarrow x \in (\exists r.E^{ii})^j. \end{aligned} \tag{5.11}$$

By definition M_i and thus also Λ_i contains $\exists r.E^{ii}$ (up to equivalence). Hence, $\mathcal{B}_2$ contains the GCI $\exists r.E^{ii} \sqsubseteq (\exists r.E^{ii})^{ii}$. Since all GCIs from $\mathcal{B}_2$ hold in j the GCI $\exists r.E^{ii} \sqsubseteq (\exists r.E^{ii})^{ii}$ must also hold in j:

$$(\exists r.E^{ii})^j \subseteq ((\exists r.E^{ii})^{ii})^j \tag{5.12}$$

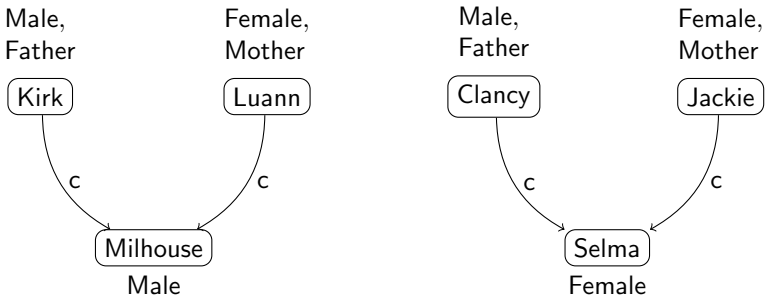

Figure 5.7: A Model Consisting of Two Families

We can apply Lemma 4.2 and obtain

$$((\exists r.E^{ii})^{ii})^j = ((\exists r.E)^{ii})^j = (D^{ii})^j. \tag{5.13}$$

(5.10), (5.11), (5.12), and (5.13) show that $x \in D^j$ implies $x \in (D^{ii})^j$, i. e. $D \sqsubseteq D^{ii}$ holds in j. We have shown that $D \sqsubseteq D^{ii}$ follows from $\mathcal{B}_2$.

It follows that every GCI from $\mathcal{B}_1$ follows from $\mathcal{B}_2$. Since $\mathcal{B}_1$ is complete for i it follows that $\mathcal{B}_2$ is also complete for i. Since $\mathcal{B}_2$ is also sound for i this implies that $\mathcal{B}_2$ is a finite base for the GCIs holding in i. □

Example 5.3. We introduce an example that we will come back to several times in this chapter and the following. Let the model i describe two families. The domain of i consists of six persons: Kirk, Luann and their son Milhouse, as well as Clancy, Jackie and their daughter Selma (cf. Figure 5.7). As primitive concept names we use Female, Male, Mother, Father and as role name we use c. In order to compute M_i we first compute all concept descriptions of the form X^i where $X \subseteq \Delta_i$, $X \neq \emptyset$. For matters of convenience, we introduce abbreviations for them, such as FoD for a concept describing a father of a daughter, PoS for "parent of son" or MoC for "mother of child". We obtain 12 model-based most specific concepts.

$$\begin{array}{ll}
\top, & \mathsf{MoC} = \mathsf{Mother} \sqcap \mathsf{Female} \sqcap \exists \mathsf{c}.\top, \\
\mathsf{Female}, & \mathsf{FoC} = \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\top, \\
\mathsf{Male}, & \mathsf{MoD} = \mathsf{Mother} \sqcap \mathsf{Female} \sqcap \exists \mathsf{c}.\mathsf{Female}, \\
\mathsf{PoC} = \exists \mathsf{c}.\top, & \mathsf{FoD} = \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\mathsf{Female}, \\
\mathsf{PoD} = \exists \mathsf{c}.\mathsf{Female}, & \mathsf{MoS} = \mathsf{Mother} \sqcap \mathsf{Female} \sqcap \exists \mathsf{c}.\mathsf{Male}, \\
\mathsf{PoS} = \exists \mathsf{c}.\mathsf{Male}, & \mathsf{FoS} = \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\mathsf{Male}.
\end{array} \tag{5.14}$$

We obtain M_i from

$$M_i = \{\bot\} \cup \{\mathsf{Mother}, \mathsf{Father}, \mathsf{Female}, \mathsf{Male}\} \cup \{\exists \mathsf{c}.X^i \mid X \subseteq \Delta_i, X \neq \emptyset\}.$$

The total number of attributes in M_i is $|M_i| = 1 + 4 + 12 = 17$. There are 2^{17} subsets of M_i and thus $|\Lambda_i| = |\mathcal{B}_2| = 2^{17}$. It can be seen that even the small model i we obtain an impractically large number of GCIs in $\mathcal{B}_2$.

In this section we have shown that $\mathcal{B}_2$ is a finite base for the GCIs holding in i. In particular, this shows that it is possible to axiomatize every model i in $\mathcal{EL}^{\bot}_{\text{gfp}}$. Our aim is to obtain a base that is as small as possible. In theory it is possible to use a brute force approach in order to obtain a minimal base from $\mathcal{B}_2$. We can look at each GCI in $\mathcal{B}_2$ and check whether it follows from $\mathcal{B}_2$. If it does follow we remove it, and if it does not follow we mark it. This process is repeated until all GCIs are marked resulting in a minimal base for the GCIs holding in i. This brute force approach is, however, not very efficient. In the worst case the size of M_i is already exponential in the size of Δ_i since there can be exponentially many subsets $X \subseteq \Delta_i$. Since Λ_i is exponential in the size of M_i the size of $\mathcal{B}_2$ is double-exponential in the size of Δ_i. We wish to have a smarter approach that does not require computing $\mathcal{B}_2$ in its entirety. This is done in Section 5.2.

On Computing M_i

So far, we have not addressed the problem of computing the set M_i itself. M_i is defined as

$$M_i = \{\bot\} \cup \mathcal{N}_C \cup \{\exists r.C \mid r \in \mathcal{N}_R, C \in \mathcal{X}_i\} \tag{5.15}$$

where

$$\mathcal{X}_i = \{X^i \mid X \subseteq \Delta_i, X \neq \emptyset\}.$$

The most costly task when computing M_i is clearly the computation of $\mathcal{X}_i$. The naive approach is to compute X^i for every possible subset $X \subseteq \Delta_i$, $X \neq \emptyset$. This computation of all subsets requires, of course, exponential time in the size of Δ_i. In this section we show how Next-Closure can be used to compute $\mathcal{X}_i$ more efficiently.

Another way to obtain all concept descriptions of the form X^i where $X \subseteq \Delta_i$, $X \neq \emptyset$, makes use of Lemma 4.1. Lemma 4.1 states that $X^i \equiv (X^{ii})^i$ holds for all $X \subseteq \Delta_i$. Thus we obtain

$$\begin{aligned}\mathcal{X}_i =& \{(X^{ii})^i \mid X \subseteq \Delta_i, X \neq \emptyset\} \\ =& \{Y^i \mid Y \in \mathcal{Y}_i\}\end{aligned}$$

where we define $\mathcal{Y}_i$ to be the set

$$\mathcal{Y}_i = \{X^{ii} \mid X \subseteq \Delta_i, X \neq \emptyset\}.$$

The idea is to show that $\mathcal{Y}_i$ is a closure system and can therefore be computed using Next-Closure. We show that $\cdot^{ii} \colon 2^{\Delta_i} \to 2^{\Delta_i}$ is a closure operator.

Lemma 5.11. *The mapping $\cdot^{ii} \colon 2^{\Delta_i} \to 2^{\Delta_i}$ is a closure operator on Δ_i.*

Proof. All three properties of closure operators are easy consequences of Lemma 4.1. *$\cdot^{ii}$ is idempotent:* It follows from Lemma 4.1 (5) that $(X^{ii})^{ii} = X^{ii}$ for all $X \subseteq \Delta_i$. *$\cdot^{ii}$ is extensive:* This follows trivially from Lemma 4.1 (3). *$\cdot^{ii}$ is order-preserving:* Let $X, Y \subseteq \Delta_i$ be two sets of individuals that satisfy $X \subseteq Y$. From Lemma 4.1 (1) we obtain $X^i \sqsubseteq Y^i$ and from Lemma 4.1 (2) we obtain $(X^i)^i \subseteq (Y^i)^i$. This proves that $\cdot^{ii}$ is a closure operator on Δ_i. □

Since $\cdot^{ii}$ is a closure operator on Δ_i we can instantiate Next-Closure in its general form (Algorithm 1) with $\cdot^{ii}$ as the closure operator in order to obtain $\mathcal{Y}_i$. Once we have computed $\mathcal{Y}_i$ we can obtain $\mathcal{X}_i$ by applying $\cdot^i$ to every set $Y \in \mathcal{Y}_i$. We then obtain M_i from $\mathcal{X}_i$ and (5.15). Algorithm 6 shows how Next-Closure can be used to compute M_i.

Algorithm 6 Computing M_i Using Next-Closure

1: **input** $i = (\Delta_i, \cdot^i)$ {model i}
2: $Y_0 := \emptyset$ {lectically first set that is closed with respect to $\cdot^{ii}$}
3: $\mathcal{X}_0 := \emptyset$, $k := 0$
4: **while** $Y_k \neq \Delta_i$ **do**
5: Y_{k+1} = lectically smallest subset of Δ_i that is
– lectically greater than Y_k, and
– closed with respect to $\cdot^{ii}$
6: $\mathcal{X}_{k+1} := \mathcal{X}_k \cup \{Y_{k+1}^i\}$
7: $k := k + 1$
8: **end while**
9: **return** $\{\bot\} \cup \mathcal{N}_C \cup \{\exists r.C \mid C \in \mathcal{X}_k\}$

5.2 Reducing the Size of the Base

Let $i = (\Delta_i, \cdot^i)$ be an $\mathcal{EL}^\bot_{\text{gfp}}$-model and $\mathcal{B}_2$ the finite base for the GCIs holding in i as defined in Theorem 5.10. It has been argued in the previous section that the base $\mathcal{B}_2$ can be very large. The aim of this section is to identify small subsets of $\mathcal{B}_2$ that are still complete. This is done in the first part of this section using induced contexts and FCA. In the second part we prove that the reduced base has minimal cardinality among all the $\mathcal{EL}^\bot_{\text{gfp}}$-bases for the GCIs of a given model.

5.2.1 Removal of Redundancy Using Induced Contexts

Our approach for axiomatizing finite models uses a combination of both ideas from Section 4. In Section 5.1.2 we have computed a finite base for a given model i using model-based most specific concepts. In this section we use this finite base as a starting point and reduce its size by applying FCA methods to an induced context.

In Section 4.2 we have seen that a set of attributes $A \subseteq M$ in an induced context $\mathbb{K}$ can be converted into a concept description by taking the conjunction $\bigsqcap A$ over the attributes in A. The elements of Λ_i are obtained as conjunctions of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions from the set M_i. Therefore it is natural to view the set M_i as the set of attributes of an induced context.

Something that we have not talked about in the section about induced contexts are logical dependencies among attributes. For example, the attribute set of the context $\mathbb{K}$ from Example 4.3 contains the attributes $\bot$ and Female. Since Female subsumes $\bot$ it would be redundant to add the GCI $\bot \sqsubseteq$ Female to the knowledge base. In order to take care of these logical dependencies among the attributes of an induced context we introduce background knowledge. Let $\mathcal{S}_{M_i}$ be the set of implications

$$\mathcal{S}_{M_i} = \{\{C\} \to \{D\} \mid C, D \in M_i, C \sqsubseteq D\} \tag{5.16}$$

We call $\mathcal{S}_{M_i}$ the *DL-background knowledge* about M_i.

Let $\mathbb{K}_i$ be the context induced by M_i and i. D subsumes C if $C \sqsubseteq D$ holds in every interpretation j. Hence, every implication from $\mathcal{S}_{M_i}$ holds in $\mathbb{K}_i$. We prove that there is a correspondence between the $\mathcal{S}_{M_i}$-bases of $\mathbb{K}_i$ and the complete subsets of $\mathcal{B}_2$.

Theorem 5.12. *Let $i = (\Delta_i, \cdot^i)$ be a model and let $\mathbb{K}_i$ be the context induced by M_i and i. Let $\mathcal{L}$ be an $\mathcal{S}_{M_i}$-base of $\mathbb{K}_i$ that contains only implications of the form $U \to U''$.*[4] *Then*

$$\mathcal{B}_3 = \{\bigsqcap U \sqsubseteq (\bigsqcap U)^{ii} \mid U \to U'' \in \mathcal{L}\}$$

is a finite base for the GCIs holding in i.

Proof. Since $\mathcal{B}_3$ is a subset of $\mathcal{B}_2$ we know that $\mathcal{B}_3$ is finite and that it is sound for i. It remains to show that $\mathcal{B}_3$ is complete for i. We show that every GCI from $\mathcal{B}_2$ follows from $\mathcal{B}_3$.

Let $j = (\Delta_j, \cdot^j)$ be a model in which all GCIs from $\mathcal{B}_3$ hold. Let $\mathbb{K}_j$ be the context induced by M_i and j. We are now dealing with two different contexts $\mathbb{K}_j$ and $\mathbb{K}_i$. To avoid confusion we will denote the

[4] If $\mathcal{L}$ contains implications of a different type it can be transformed into a $\mathcal{S}_{M_i}$-base with the same number of implications that contains only implications of the form $U \to U''$ using Lemma 3.4.

derivation operators from $\mathbb{K}_i$ and $\mathbb{K}_j$ by $\cdot'^i$ and $\cdot'^j$, respectively. The idea of the proof is to first show that all implications from $\mathcal{L}$ and $\mathcal{S}_{M_i}$ hold in $\mathbb{K}_j$ also. From this we conclude that all implications of the form $V \to V''^i$, where V is a subset of M_i, hold in $\mathbb{K}_j$. Last, we derive that $\bigcap V \sqsubseteq (\bigcap V)^{ii}$ holds in j.

We first establish a connection between the derivation operators in $\mathbb{K}_j$ and the interpretation j. Let $U \subseteq M_i$ be a set of attributes in $\mathbb{K}_j$. Then Lemma 4.10 shows that U satisfies

$$(\bigcap U)^j = U'^j. \tag{5.17}$$

We know from Lemma 5.9 that $(\bigcap U)^{ii}$ can be expressed in terms of M_i. From Lemma 4.11 we obtain

$$\big((\bigcap U)^{ii}\big)^j = \big(\mathrm{pr}_{M_i}((\bigcap U)^{ii})\big)'^j.$$

Corollary 4.13 proves $\mathrm{pr}_{M_i}((\bigcap U)^{ii}) = U''^i$. Hence, all subsets $U \subseteq M_i$ satisfy

$$\big((\bigcap U)^{ii}\big)^j = \big(U''^i\big)'^j. \tag{5.18}$$

We show that all implications from $\mathcal{L}$ hold in $\mathbb{K}_j$. Let $U \to U''^i$ be an implication from $\mathcal{L}$. Since $\bigcap U \sqsubseteq (\bigcap U)^{ii}$ holds in j we obtain

$$(\bigcap U)^j \subseteq \big((\bigcap U)^{ii}\big)^j.$$

(5.17) and (5.18) yield

$$U'^j \subseteq \big(U''^i\big)'^j.$$

Hence, all implications from $\mathcal{L}$ hold in $\mathbb{K}_j$. The implications from $\mathcal{S}_{M_i}$ hold in every induced context with the attribute set M_i.

Let $\bigcap V \sqsubseteq (\bigcap V)^{ii}$, where $V \subseteq M_i$, be a GCI from $\mathcal{B}_2$. Clearly, $V \to V''^i$ holds in the context $\mathbb{K}_i$. Since $\mathcal{L}$ is an $\mathcal{S}_{M_i}$-base for $\mathbb{K}_i$ the implication $V \to V''^i$ follows from $\mathcal{L}$ and $\mathcal{S}_{M_i}$. We have shown that all implications from $\mathcal{L}$ and $\mathcal{S}_{M_i}$ hold in $\mathbb{K}_j$, thus $V \to V''^i$ also holds in $\mathbb{K}_j$. This yields

$$V'^j \subseteq \big(V''^i\big)'^j.$$

Using (5.17) and (5.18) again, we obtain that

$$(\bigcap V)^j \subseteq \big((\bigcap V)^{ii}\big)^j,$$

i. e. $\bigcap V \sqsubseteq (\bigcap V)^{ii}$ holds in j. Therefore we have shown that every GCI from $\mathcal{B}_2$ follows from $\mathcal{B}_3$. Since $\mathcal{B}_2$ is complete $\mathcal{B}_3$ is also complete for i and since $\mathcal{B}_3$ is also sound it is a base for the GCIs holding in i. □

Corollary 5.13. *Let $i = (\Delta_i, \cdot^i)$ be a model and let $\mathbb{K}_i$ be the context induced by M_i and i. Let $\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}$ be the $\mathcal{S}_{M_i}$-Duquenne-Guigues Base of $\mathbb{K}_i$. Then*

$$\mathcal{B}_{\mathcal{DG}} = \{\bigcap U \sqsubseteq (\bigcap U)^{ii} \mid U \to U'' \in \mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}\}$$

is a finite base for the GCIs holding in i.

We have mentioned earlier that all results from this section also hold, if we replace M_i by a set of concept descriptions $\bar{M}$ that is equal up to equivalence. Let $\bar{M}$ be a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions that satisfies $\bar{M} \doteq M_i$. We define $\bar{\mathcal{S}}$ to be the set

$$\bar{\mathcal{S}} = \{\{C\} \to \{D\} \mid C, D \in \bar{M}, C \sqsubseteq D\}.$$

Corollary 5.14. *Let $i = (\Delta_i, \cdot^i)$ be a model Define $\bar{\mathbb{K}}$ as the context induced by $\bar{M}$ and i. If $\mathcal{L}$ is a $\bar{\mathcal{S}}$-base of $\bar{\mathbb{K}}$ that contains only implications of the form $U \to U''$ then*

$$\mathcal{B}_3 = \{\bigcap U \sqsubseteq (\bigcap U)^{ii} \mid U \to U'' \in \mathcal{L}\}$$

is a finite base for the GCIs holding in i.

Example 5.4. Let us get back to the model from Example 5.3 (cf. Figure 5.7). The set of attributes M_i is

$$M_i = \{\bot\} \cup \{\mathsf{Mother}, \mathsf{Father}, \mathsf{Female}, \mathsf{Male}\} \cup \{\exists \mathsf{c}.X^i \mid X \subseteq \Delta_i, X \neq \emptyset\}.$$

The possible values for X^i, where $X \subseteq \Delta_i$, $X \neq \emptyset$, are listed in (5.14). Table 5.1 shows the context $\mathbb{K}_i$ induced by M_i and i. In order to obtain the DL-background knowledge about M_i the whole set M_i has to be classified. The result of the classification is shown in Figure 5.8. $\mathcal{S}_{M_i}$ contains an implication $\{C\} \to \{D\}$ if it is possible to move upwards along edges from C to D in Figure 5.8. The $\mathcal{S}_{M_i}$-pseudo-intents of $\mathbb{K}_i$ can be obtained using Algorithm 5. They are the seven sets $\{\mathsf{Female}, \mathsf{Male}\}$, $\{\mathsf{Mother}\}$, $\{\mathsf{Father}\}$, $\{\mathsf{Female}, \exists \mathsf{c}.\top\}$, $\{\mathsf{Male}, \exists \mathsf{c}.\top\}$, $\{\exists \mathsf{c}.\top, \exists \mathsf{c}.\mathsf{Female}, \exists \mathsf{c}.\mathsf{Male}\}$ and $\{\exists \mathsf{c}.\top, \exists \mathsf{c}.\mathsf{PoC}\}$. The $\mathcal{S}_{M_i}$-pseudo-intents yield the following GCIs

Table 5.1: The Induced Context $\mathbb{K}_i$ Corresponding to the Model from Figure 5.7

	$\bot$	Female	Male	Mother	Father	$\exists c.\top$	$\exists c.$Female	$\exists c.$Male	$\exists c.$PoC	$\exists c.$PoD	$\exists c.$PoS	$\exists c.$MoC	$\exists c.$FoC	$\exists c.$MoD	$\exists c.$FoD	$\exists c.$MoS	$\exists c.$FoS
Kirk			×		×	×		×									
Luann		×		×		×		×									
Milhouse			×														
Clancy			×		×	×	×										
Jackie		×		×		×	×										
Selma		×															

- Female $\sqcap$ Male $\sqsubseteq \bot$,
- Mother $\sqsubseteq$ Female $\sqcap \exists c.\top$,
- Father $\sqsubseteq$ Male $\sqcap \exists c.\top$,
- Female $\sqcap \exists c.\top \sqsubseteq$ Mother,
- Male $\sqcap \exists c.\top \sqsubseteq$ Father,
- $\exists c.$Male $\sqcap \exists c.$Female $\sqsubseteq \bot$, and
- $\exists c.\exists c.\top \sqsubseteq \bot$.

This demonstrates that using induced contexts and background knowledge we can reduce the size of the base $\mathcal{B}_2$, which is 2^{17} for this model, to only 6 GCIs in $\mathcal{B}_{\mathcal{DG}}$. The total number of attributes is only 17. 17 attributes may appear to be a large number, but one has to consider that there are 256 possible $\mathcal{EL}$-concept description of role depth less than or equal to 2 over the signature of i counting only those that cannot be written as a conjunction (We need to compute GCIs at least up to role depth 2 since 2 is the greatest role depth that occurs as a left-hand side $\mathcal{B}_{\mathcal{DG}}$). If we naively increase the role depth of attributes in the induced context we have to add all of these 256 attributes. Thus, we see a tremendous improvement with respect to the size of the context compared to this naive approach.

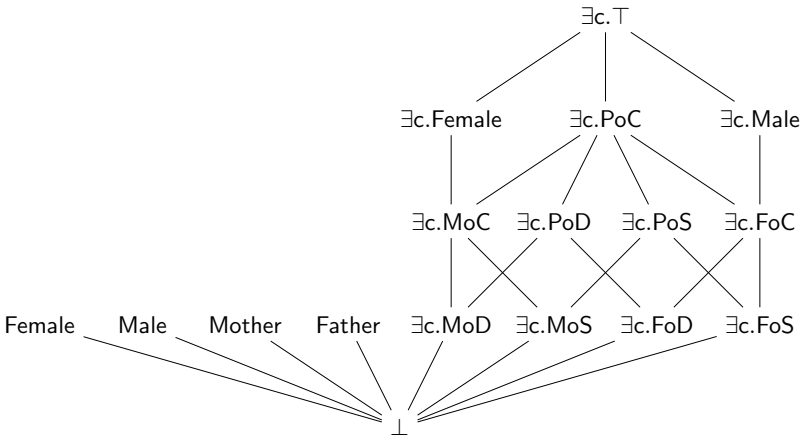

Figure 5.8: Background Knowledge

5.2.2 Minimal Cardinality

Example 5.4 demonstrates that $\mathcal{B}_{\mathcal{DG}}$ is small compared to the bases that we have seen before. In this section we demonstrate that it even has minimal cardinality among all bases for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-bases holding in i. Our proof makes use of Theorem 3.8, which states that the $\mathcal{S}$-Duquenne-Guigues Base $\mathcal{DG}^{\mathcal{S}}_{\mathbb{K}}$ is a minimal cardinality $\mathcal{S}$-base of $\mathbb{K}$. One of the arguments in the proof of Theorem 3.8 uses the following property of implications in FCA. Let $\mathbb{K} = (G, M, I)$ be a formal context, $U, V \subseteq M$ sets of attributes, where $V \not\subseteq U$, and $\mathcal{L}$ a set of implications. Then it holds that

$$U \to V \text{ follows from } \mathcal{L} \Longrightarrow \exists (A \to B) \in \mathcal{L} : A \subseteq U \text{ and } B \not\subseteq U \quad (5.19)$$

In other words if there is a non-trivial implication $U \to V$ that follows from $\mathcal{L}$, then U does not respect all implications from $\mathcal{L}$. This property cannot be translated to the $\mathcal{EL}^{\bot}_{\text{gfp}}$-setting by naively replacing implications by GCIs and replacing the subset relation $\subseteq$ by the subsumption relation $\sqsupseteq$. For example the GCI

$$\exists \mathsf{marriedTo}.\mathsf{Parent} \sqsubseteq \exists \mathsf{marriedTo}.\exists \mathsf{hasChild}.\top$$

follows from the set of GCIs $\mathcal{B} = \{\mathsf{Parent} \sqsubseteq \exists \mathsf{hasChild}.\top\}$, even though $\mathcal{B}$ does not contain an implication $C \sqsubseteq D$ satisfying $C \sqsubseteq \exists \mathsf{marriedTo}.\mathsf{Parent}$ and $D \not\sqsubseteq \exists \mathsf{marriedTo}.\mathsf{Parent}$. Inferences in $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$ can take place "behind" existential quantifiers.[5]

The set Λ_i as defined in Definition 5.3 contains only concept descriptions where every quantifier is followed by a model based most specific concept. Every description $C \in \Lambda_i$, $C \neq \bot$, can be written as

$$C \equiv \bigsqcap U \sqcap \bigsqcap_{(r,Y)\in\Pi} \exists r.Y^i,$$

for a set of concept names $U \subseteq \mathcal{N}_C$ and a set of pairs $\Pi \subseteq \mathcal{N}_R \times 2^{\Delta_i}$. For the special case of descriptions of this form we can find a result that is similar to (5.19). Remember that model based most specific concepts Y^i can be obtained as the least common subsumer of all concept descriptions $\{y\}^i$, $y \in Y$ (Lemma 4.6). The least common subsumer is obtained by forming the product of the $\mathcal{EL}$-description graphs of the descriptions $\{y\}^i$ (Lemma 2.4). By Lemma 4.5 the description graphs of $\{y\}^i$ are identical to the description graphs of i, and hence G_{Y^i} is the product of $n = |Y|$ instances of the description graph G_i of i. We prove a technical result about the product of $\mathcal{EL}$-description graphs.

Lemma 5.15. *Let $i = (\Delta_i, \cdot^i)$ be an $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-interpretation and G_i its $\mathcal{EL}$-description graph. Let n be a natural number and $G_n = G_i \otimes \cdots \otimes G_i$ be the product of n instances of G_i. Define $i_n = (\Delta_n, \cdot^{i_n})$ to be the model that can be obtained from G_n using Definition 2.22. Then every $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-GCI that holds in i also holds in i_n.*

Proof. We only give an outline of the proof for the case where $C \neq \bot$ and $D \neq \bot$. The case where $D = \bot$ uses a similar argument and the case where $C = \bot$ is trivial. Let $C \sqsubseteq D$, where $C = (A_C, \mathcal{T}_C)$ and $D = (A_D, \mathcal{T}_D)$, be a GCI that holds in i. Since i_n is obtained from G_n it holds that $\Delta_{i_n} = \Delta_i \times \cdot \times \Delta_i$. Let $(x_1, \ldots, x_n) \in C^{i_n}$ be an individual in the extension of C in i_n. The outline of the proof is as follows: One first shows that $(x_1, \ldots, x_n) \in C^{i_n}$ implies that the individuals x_k, for all $1 \leq k \leq n$, are in C^i. Since $C \sqsubseteq D$ holds in i this implies $x_k \in D^i$ for all $1 \leq k \leq n$. Last we derive $(x_1, \ldots, x_k) \in D^{i_n}$. To prove $x_k \in C^i$

[5] In a rule based calculus this must be taken care of by introducing an $\exists$-lifting rule as in Definition 3.3 of [Rud06]

and to prove $(x_1, \ldots, x_k) \in D^{i_n}$ we construct simulations and apply Lemma 2.2. We only give the formal definitions of the corresponding simulations and omit the technical verification of properties (S1) and (S2).

Lemma 2.2 and $(x_1, \ldots, x_k) \in C^{i_n}$ yield that there is a simulation Z_1 from $G_{\mathcal{T}_C}$ to G_n that contains $(A_C, (x_1, \ldots, x_n))$. For each k, $1 \leq k \leq n$, we define a relation

$$Z_1^k = \big\{(B, y) \mid \exists y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n \colon (B, (y_1, \ldots, y_{k-1}, x_k, y_{k+1}, \ldots, y_n)) \in Z_1\big\}.$$

It can be shown that for all k the relation Z_1^k is a simulation from $G_{\mathcal{T}_C}$ to G_i containing (A_C, x_k). Then Lemma 2.2 proves that $(x_1, \ldots, x_n) \in C^{i_n}$ implies $x_k \in C^i$ for all $k \in \{1, \ldots, n\}$.

Now assume that $x_k \in D^i$ holds for all $1 \leq k \leq n$. Lemma 2.2 implies that for each k, $1 \leq k \leq n$, there is a simulation Z_2^k from $G_{\mathcal{T}_D}$ to G_i that contains (A_D, x_k). Define

$$Z_2 = \{(B, (y_1, \ldots, y_n)) \mid \forall 1 \leq k \leq n \colon (B, y_n) \in Z_2^k\}.$$

One can show that Z_2 is a simulation from $G_{\mathcal{T}_D}$ to G_n that contains $(A_D, (x_1, \ldots, x_k))$. Lemma 2.2 proves that $x_k \in D^i$ for all $k \in \{1, \ldots, n\}$ implies $(x_1, \ldots, x_n) \in D^{i_n}$. □

Lemma 5.16. *Let $i = (\Delta_i, \cdot^i)$ be an interpretation and $\mathcal{B}$ a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs that hold in i. Let $C = (A_C, \mathcal{T}_C) \in \Lambda_i$ be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description from Λ_i and let $D = (A_D, \mathcal{T}_D)$ be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description such that $C \not\sqsubseteq D$. Then it holds that*

$$C \sqsubseteq D \text{ follows from } \mathcal{B} \Longrightarrow \exists (E \sqsubseteq F) \in \mathcal{B} \colon C \sqsubseteq E \text{ and } C \not\sqsubseteq F. \quad (5.20)$$

Proof. We know that $C \neq \bot$ because this would contradict $C \not\sqsubseteq D$. Since $C \in \Lambda_i$ holds C can be written as

$$C \equiv \bigsqcap U \sqcap \bigsqcap_{(r,Y) \in \Pi} \exists r.Y^i.$$

for some set of concept names $U \subseteq \mathcal{N}_C$ and some set of pairs $\Pi \subseteq \mathcal{N}_R \times 2^{\Delta_i}$.

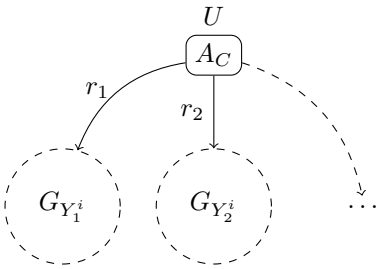

Figure 5.9: Description Graph of $C \in \Lambda_i$

Let G_C be the description graph of C. The graph G_C consists of a vertex A_C which is connected by an r-edge to the description graph G_{Y^i} of Y^i for each pair $(r, Y) \in \Pi$ (Figure 5.9).

Let $i_C = (\Delta_{i_C}, \cdot^{i_C})$, where $\Delta_{i_C} = \mathcal{N}_D(\mathcal{T}_C)$ be the model that is obtained from G_C according to Definition 2.22. In our proof we proceed as follows. First we show that i_C cannot be a model of $\mathcal{B}$, and thus there must be a GCI $E \sqsubseteq F$ from $\mathcal{B}$ that does not hold in i_C. Using the technical Lemma 5.15 we can show that the parts of i_C that correspond to some G_{Y^i} cannot contain a counterexample to $E \sqsubseteq F$, and therefore the counterexample has to be the individual that corresponds to the root concept A_C. Translating the model i_C back into the concept description then yields the claim.

We know that $C \not\sqsubseteq D$ holds. Therefore Lemma 2.3 implies that there is no simulation from G_D to G_C containing (A_D, A_C). Lemma 2.2 shows that A_C viewed as an individual in Δ_{i_C} is not an element of D^{i_C}. Hence, $C \sqsubseteq D$ does not hold in i_C. Because $C \sqsubseteq D$ follows from $\mathcal{B}$ there must be some GCI $E \sqsubseteq F$ from $\mathcal{B}$ that does not hold in i_C. Thus, there is an individual $x \in \Delta_{i_C}$ with $x \in E^{i_C}$ but $x \notin F^{i_C}$.

Each description graph G_{Y^i} is the product of $n = |Y|$ instances of the description graph G_i of i. We know that all implications from $\mathcal{B}$ hold in i. Lemma 5.15 implies that all implications from $\mathcal{B}$ hold in all models that correspond to G_{Y^i} for $(r, Y) \in \Pi$. Thus $y \neq A_C$ and $y \in E^{i_C}$ implies $y \in F^{i_C}$. This shows that $x = A_C$.

Because $x = A_C \in E^{i_C}$ holds Lemma 2.2 shows that there is a simulation from E to A_C in i_C. Since i_C and $\mathcal{T}_C$ have the same description

graph Z must also be a simulation from E to C. By Lemma 2.3 this proves $C \sqsubseteq E$. Analogously we can derive from $x = A_C \notin F^{i_C}$ that $C \not\sqsubseteq F$. □

Lemma 5.17. *Let C be an $\mathcal{EL}_{\text{gfp}}$-concept description, $U \subseteq M_i$ a subset of M_i. Then $\bigsqcap U \sqsubseteq C$ implies $\bigsqcap U \sqsubseteq \text{approx}_i(C)$.*

Proof. C can be written as

$$C \equiv \bigsqcap S \sqcap \bigsqcap_{(r,D)\in\Pi} \exists r.D.$$

for some set of concept names $S \subseteq \mathcal{N}_C$ and some set of pairs Π of role names and $\mathcal{EL}_{\text{gfp}}$-concept descriptions. The approximation $\text{approx}_i(C)$ is defined as

$$C \equiv \bigsqcap S \sqcap \bigsqcap_{(r,D)\in\Pi} \exists r.D^{ii}.$$

$\bigsqcap U \sqsubseteq C$ yields that $A \in S$ implies $A \in U$. Likewise, $\bigsqcap U \sqsubseteq C$ implies that for every concept description $\exists r.D$, where $(r, D) \in \Pi$, there must be some description $\exists r.X^i \in U$ that satisfies $\exists r.X^i \sqsubseteq \exists r.D$. Then X also satisfies $X^i \sqsubseteq D$. Lemma 4.1 yields $X^i \equiv X^{iii} \sqsubseteq D^{ii}$ and $\exists r.X^i \sqsubseteq \exists r.D^{ii}$ follows. We therefore obtain

$$\bigsqcap U \sqsubseteq \bigsqcap S \sqcap \bigsqcap_{(r,D)\in\Pi} \exists r.D^{ii}.$$

Thus, $\bigsqcap U \sqsubseteq \text{approx}_i(C)$ holds. □

Theorem 5.18. *Let $i = (\Delta_i, \cdot^i)$ be a finite model and let $\mathcal{B}_{\mathcal{DG}}$ be defined as in Corollary 5.13. Then the base $\mathcal{B}_{\mathcal{DG}}$ has minimal cardinality among all bases for the GCIs holding in i.*

Proof. Assume that $\mathcal{B}$ is some finite base for the GCIs that hold in i. We show that $\mathcal{B}$ contains at least as many GCIs as $\mathcal{B}_{\mathcal{DG}}$. Without loss of generality we can assume that $\mathcal{B}$ contains only GCIs of the form $E \sqsubseteq E^{ii}$. The number of GCIs in $\mathcal{B}_{\mathcal{DG}}$ equals the number of implications in $\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}$, where $\mathbb{K}_i$ is the context induced by M_i and i, $\mathcal{S}_{M_i}$ is the DL-background knowledge about M_i and $\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}$ is the $\mathcal{S}_{M_i}$-Duquenne Guigues Base for $\mathbb{K}_i$.

$$|\mathcal{B}_{\mathcal{DG}}| = |\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}| \tag{5.21}$$

We define $\mathcal{L}_\mathcal{B}$ to be the set of implications

$$\mathcal{L}_\mathcal{B} = \{\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \to \mathrm{pr}_{M_i}(E^{ii}) \mid (E \sqsubseteq E^{ii}) \in \mathcal{B}\}.$$

Clearly, $\mathcal{L}_\mathcal{B}$ contains at most as many implications as there are GCIs in $\mathcal{B}$.

$$|\mathcal{L}_\mathcal{B}| \leq |\mathcal{B}| \tag{5.22}$$

Therefore in order to prove that $\mathcal{B}$ contains at least as many GCIs as $\mathcal{B}_{\mathcal{DG}}$, it suffices to show that $\mathcal{L}_\mathcal{B}$ contains at least as many implications as $\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}$. To prove

$$|\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}| \leq |\mathcal{L}_\mathcal{B}| \tag{5.23}$$

it suffices to show that $\mathcal{L}_\mathcal{B}$ is an $\mathcal{S}_{M_i}$-base for $\mathbb{K}_i$. Then (5.23) follows from Theorem 3.8.

Soundness: Let $\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \to \mathrm{pr}_{M_i}(E^{ii})$ be an implication from $\mathcal{L}_\mathcal{B}$. Then $\mathcal{B}$ contains $E \sqsubseteq E^{ii}$. Since $\mathrm{approx}_i(E)$ can be expressed in terms of M_i it follows from Lemma 4.11 that

$$\big(\mathrm{pr}_{M_i}(\mathrm{approx}_i(E))\big)' = (\mathrm{approx}_i(E))^i$$

holds. Lemma 5.8 yields $\mathrm{approx}_i(E) \sqsubseteq E$ and thus Lemma 4.1 implies

$$(\mathrm{approx}_i(E))^i \subseteq E^i \equiv (E^{ii})^i.$$

Since E^{ii} can be expressed in terms of M_i we obtain

$$(E^{ii})^i = \big(\mathrm{pr}_{M_i}(E^{ii})\big)'.$$

from Lemma 4.11. We have thus shown that $\big(\mathrm{pr}_{M_i}(\mathrm{approx}_i(E))\big)' \subseteq \big(\mathrm{pr}_{M_i}(E^{ii})\big)'$ holds. Therefore $\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \to \mathrm{pr}_{M_i}(E^{ii})$ holds in $\mathbb{K}_i$.

Completeness: To prove completeness of $\mathcal{L}_\mathcal{B} \cup \mathcal{S}_{M_i}$ it suffices to show that no set $U \subseteq M_i$ with $U \neq U''$ respects all implications from $\mathcal{L}_\mathcal{B} \cup \mathcal{S}_{M_i}$.

Let $U \subseteq M_i$ be a subset of M_i with $U \neq U''$. Assume that U respects all implications from $\mathcal{S}_{M_i}$. $U \neq U''$ implies that there is some attribute $D \in M_i \setminus U$ such that all $x \in U'$ satisfy xID, i. e. all $x \in U'$ satisfy $x \in D^i$. This implies $U' \subseteq D^i$. Lemma 4.10 implies that $U' = (\bigsqcap U)^i$ holds. We obtain $(\bigsqcap U)^i \subseteq D^i$. Therefore, Lemma 4.1 yields

$$(\bigsqcap U)^{ii} \sqsubseteq D. \tag{5.24}$$

Because U respects all implications from $\mathcal{S}_{M_i}$ and because of $D \notin U$ we obtain that $F \not\sqsubseteq D$ for all $F \in U$. Since D is from M_i and thus either a concept name or of the form $\exists r.X^i$ for some $r \in \mathcal{N}_R$ and some $X \subseteq \Delta_i$ it follows that

$$\bigsqcap U \not\sqsubseteq D \tag{5.25}$$

holds. (5.24) and (5.25) imply

$$\bigsqcap U \not\sqsubseteq (\bigsqcap U)^{ii}. \tag{5.26}$$

Lemma 5.16 and (5.26) show that there is some GCI $E \sqsubseteq E^{ii} \in \mathcal{B}$ such that $\bigsqcap U \sqsubseteq E$ while $\bigsqcap U \not\sqsubseteq E^{ii}$. We show that U does not respect $\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \to \mathrm{pr}_{M_i}(E^{ii})$. From Lemma 5.17 and $\bigsqcap U \sqsubseteq E$ we obtain $\bigsqcap U \sqsubseteq \mathrm{approx}_i(E)$. Projections are antitone, which implies

$$\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \subseteq U.$$

Assume that $\mathrm{pr}_{M_i}(E^{ii}) \subseteq U$ and therefore $\bigsqcap U \sqsubseteq \bigsqcap \mathrm{pr}_{M_i}(E^{ii})$ holds. Because E^{ii} can be expressed in terms of M_i it holds that $\bigsqcap \mathrm{pr}_{M_i}(E^{ii}) \equiv E^{ii}$. This contradicts $\bigsqcap U \not\sqsubseteq E^{ii}$. Therefore the assumption $\mathrm{pr}_{M_i}(E^{ii}) \subseteq U$ must be false. Thus U does not respect the implication

$$\mathrm{pr}_{M_i}(\mathrm{approx}_i(E)) \to \mathrm{pr}_{M_i}(E^{ii}).$$

We have thus shown that for every $U \subseteq M_i$ with $U \neq U''$ there is either an implication from $\mathcal{S}_{M_i}$ that is not respected by U or there is an implication from $\mathcal{L}_{\mathcal{B}}$ that is not respected by U. Therefore $\mathcal{L}_{\mathcal{B}} \cup \mathcal{S}_{M_i}$ is complete for $\mathbb{K}_i$. Since it is also sound it is an $\mathcal{S}_{M_i}$-base for $\mathbb{K}_i$. Therefore, (5.23) follows from Theorem 3.8, which states that $\mathcal{DG}_{\mathbb{K}_i}^{\mathcal{S}_{M_i}}$ has minimal cardinality among all the $\mathcal{S}_i$-bases for $\mathbb{K}_i$. From (5.21), (5.22) and (5.23) we obtain that $\mathcal{B}$ contains at least as many GCIs as $\mathcal{B}_{\mathcal{DG}}$. □

In this section we have shown that $\mathcal{B}_{\mathcal{DG}}$ has minimal cardinality among all the $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-bases of a given model. This demonstrates that our choice of the attribute set M_i combined with the FCA theory for formal contexts with background knowledge is efficient.

5.3 Obtaining an $\mathcal{EL}^{\bot}$-Base from an $\mathcal{EL}^{\bot}_{\text{gfp}}$-Base

From the previous sections we know how to compute a finite base $\mathcal{B}_3$ for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in a finite model i using induced contexts and formal concept analysis. This finite base $\mathcal{B}_3$ is at the same time a general TBox that has i as its model. The $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions that occur in the GCIs from $\mathcal{B}_3$ contain TBoxes themselves. While the general TBox $\mathcal{B}_3$ uses descriptive semantics, the TBoxes within the $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions are cyclic TBoxes with greatest-fixpoint semantics. The result is a TBox which does not stay within the limits of $\mathcal{EL}^{++}$.[6] Since $\mathcal{EL}^{++}$ is more widely used we would like to modify the base $\mathcal{B}_3$ such that its GCIs can be added to an existing $\mathcal{EL}^{++}$-knowledge base. To this purpose, we need to find a base that uses only acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions, since acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions are equivalent to $\mathcal{EL}^{\bot}$-concept descriptions.

Our approach uses the unravellings that have been introduced in Section 5.1.1. Let i be a model and let $\mathcal{B}$ be a finite $\mathcal{EL}^{\bot}_{\text{gfp}}$-base for i. We construct a finite base that uses only acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions as follows. Let $\mathcal{B}_4$ be the set of GCIs

$$\begin{aligned}\mathcal{B}_4 =&\{C_d \sqsubseteq (C^{ii})_d \mid C \sqsubseteq D \in \mathcal{B}\} \cup \\ &\{(X^i)_d \sqsubseteq (X^i)_{d+1} \mid X \subseteq \Delta_i, X \neq \emptyset\},\end{aligned} \tag{5.27}$$

where d is defined as in Lemma 5.5, and where C_d denotes the unravelling of C up to role depth d. The proof that this is a base for i requires the following simple property for unravellings.

Lemma 5.19. *Let $C = (A_C, \mathcal{T}_C)$ and $D = (A_D, \mathcal{T}_D)$ be $\mathcal{EL}_{\text{gfp}}$-concept descriptions. Then it holds that*

- $(\exists r.C)_d \equiv \exists r.C_{d-1}$, *and*
- $(C \sqcap D)_d \equiv C_d \sqcap D_d$,

where $(\exists r.C)_d$, C_d, $(C \sqcap D)_d$ and D_d denote the unravelling of $\exists r.C$, C, $C \sqcap D$ and D up to role depth d, respectively.

[6] cf. Section 8.3 for variants of $\mathcal{EL}$ that allow for both cyclic and descriptive semantics

Proof. This result is relatively easy to see, but becomes tedious if we want to prove it formally using simulations. We only give a rough outline for the proof of the first result. We need to construct a simulation from the description graph $(\exists r.C)_d$ to $\exists r.C_{d-1}$ and vice versa. We first analyze what the sets of defined concept names in $(\exists r.C)_d$ and $\exists r.C_{d-1}$ are. By definition $\exists r.C = (A_{\exists r.C}, \mathcal{T}_{\exists r.C})$, where

$$\mathcal{T}_{\exists r.C} = \{A_{\exists r.C} \equiv \exists r.A_C\} \cup \mathcal{T}_C.$$

The defined concept names of $\mathcal{T}^d_{\exists r.C}$ are the paths in the description graph of $\mathcal{T}_{\exists r.C}$ that start in $A_{\exists r.C}$ whose length is bounded by d. These paths are of the form $A_{\exists r.C} r\mathbf{w}$ where $\mathbf{w}$ is a path in the description graph of $\mathcal{T}_C$ whose length is bounded by $d-1$.

$$\mathcal{N}_D(\mathcal{T}^d_{\exists r.C}) = \{A_{\exists r.C}\} \cup \{A_{\exists r.C} r\mathbf{w} \mid \mathbf{w} \text{ path in } G_{\mathcal{T}_C} \text{ with length}(\mathbf{w}) \leq d\}$$

The concept description $\exists r.C_{d-1}$ is defined to be the pair $\exists r.C_{d-1} = (A_{\exists r.C_{d-1}}, \mathcal{T}_{\exists r.C_{d-1}})$ where

$$\mathcal{T}_{\exists r.C_{d-1}} = \{A_{\exists r.C_{d-1}} \equiv \exists r.A_C\} \cup \mathcal{T}_C^{d-1}$$

The defined concept names in $\mathcal{T}_{\exists r.C_{d-1}}$ are $A_{\exists r.C_{d-1}}$ and the defined concept names from T_C^{d-1}. Hence,

$$\begin{aligned}\mathcal{N}_D(\mathcal{T}_{\exists r.C_{d-1}}) = {} & \{A_{\exists r.C_{d-1}}\} \\ & \cup \{\mathbf{w} \mid \mathbf{w} \text{ path in } G_{\mathcal{T}_C} \text{with length}(\mathbf{w}) \leq d-1\}\end{aligned}$$

The relation

$$\begin{aligned}Z = \{(A_{\exists r.C}, A_{\exists r.C_{d-1}})\} \cup \big\{ & (A_{\exists r.C} r\mathbf{w}', \mathbf{w}') \mid \mathbf{w}' \text{ path in } G_{\mathcal{T}_C} \\ & \text{starting in } A_C \text{ of length less than } d\big\}\end{aligned}$$

is a simulation from $(\exists r.C)_d$ to $\exists r.C_{d-1}$. We omit the proof since it is very similar to many earlier proofs. Lemma 2.3 yields $\exists r.C_{d-1} \sqsubseteq (\exists r.C)_d$. Analogously, one can show that $(\exists r.C)_d \sqsubseteq \exists r.C_{d-1}$ using the inverse of Z as the simulation. This proves $(\exists r.C)_d \equiv \exists r.C_{d-1}$. $(C \sqcap D)_d \equiv C_d \sqcap D_d$ can be shown analogously. □

It has been mentioned in Section 5.2.2 that inference in $\mathcal{EL}^{\bot}_{\text{gfp}}$ can take place "behind an existential quantifier". A special case of such an inference is the following easy and well-know fact from description logics.

Lemma 5.20. *Let C and D be $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions and $r \in \mathcal{N}_R$ a role name. Then $\exists r.C \sqsubseteq \exists r.D$ follows from $\{C \sqsubseteq D\}$.*

Proof. Let $i = (\Delta_i, \cdot^i)$ be a model of $\{C \sqsubseteq D\}$. Let $x \in (\exists r.C)^i$ be an individual. By definition of the semantics of the existential quantifier this implies that there is some individual $y \in C^i$ such that $(x, y) \in r^i$. Since i is a model of $\{C \sqsubseteq D\}$ it follows that $C^i \subseteq D^i$ and hence $y \in D^i$. This means that $x \in (\exists r.D)^i$. We have shown that $(\exists r.C)^i \subseteq (\exists r.D)^i$ holds in i. Thus $\exists r.C \sqsubseteq \exists r.D$ follows from $\{C \sqsubseteq D\}$. □

Theorem 5.21. *If $\mathcal{B}$ is a finite base for the model i then $\mathcal{B}_4$ as defined in (5.27) is a finite base for i that contains only acyclic concept descriptions.*

Proof. The idea is to show that every GCI of the form $(X^i)_d \sqsubseteq X^i$ follows from the second part of $\mathcal{B}_4$, i. e. from $\mathcal{B}'_4 = \{(X^i)_d \sqsubseteq (X^i)_{d+1} \mid X \subseteq \Delta_i, X \neq \emptyset\}$.

We first show a weaker result, namely that every GCI of the form $(X^i)_k \sqsubseteq (X^i)_{k+1}$ where $k \geq d$ and where $X \subseteq \Delta_i$, $X \neq \emptyset$ follows from $\mathcal{B}'_4$. We prove this by induction over k.

Base Case: The claim holds trivially for $k = d$. *Step Case:* Assume that $(X^i)_l \sqsubseteq (X^i)_{l+1}$ follows from $\mathcal{B}_4{}'$ for all $l \in \{d, \ldots, k-1\}$. We know from Lemma 5.9 that for every $X \subseteq \Delta_i$ the description X^i is expressible in terms of M_i. Since $X \neq \emptyset$, and thus $X^i \neq \bot$, there must be a set of concept names $U \subseteq \mathcal{N}_C$ and a set of pairs $\Pi \subseteq \mathcal{N}_R \times 2^{\Delta_i}$ such that

$$X^i \equiv \bigsqcap U \sqcap \bigsqcap_{(r,Y)\in\Pi} \exists r.Y^i.$$

Lemma 5.19 proves

$$(X^i)_k \equiv \bigsqcap U \sqcap \bigsqcap_{(r,Y)\in\Pi} \exists r.(Y^i)_{k-1}. \tag{5.28}$$

By induction hypothesis $(Y^i)_{k-1} \sqsubseteq (Y^i)_k$ follows from $\mathcal{B}'_4$. Lemma 5.20 implies that

$$\bigsqcap U \sqcap \bigsqcap_{(r,Y)\in\Pi} \exists r.(Y^i)_{k-1} \sqsubseteq \bigsqcap U \sqcap \bigsqcap_{(r,Y)\in\Pi} \exists r.(Y^i)_k. \tag{5.29}$$

follows from $\mathcal{B}_4'$. Using Lemma 5.19 again, we obtain

$$\begin{aligned} \prod U \sqcap \prod_{(r,Y)\in\Pi} \exists r.(Y^i)_k &\equiv \left(\prod U \sqcap \prod_{(r,Y)\in\Pi} \exists r.Y^i \right)_{k+1} \\ &\equiv (X_i)_{k+1}. \end{aligned} \tag{5.30}$$

(5.28), (5.29) and (5.30) show that $(X^i)_k \sqsubseteq (X^i)_{k+1}$ follows from $\mathcal{B}_4'$.

The next step is to show that all GCIs of the form $(X^i)_d \sqsubseteq X^i$ follow from $\mathcal{B}_4'$. The case where $X^i = \bot$ is trivial. Therefore, we can assume $X \neq \emptyset$. Let $j = (\Delta_j, \cdot^j)$ be an interpretation in which all GCIs from $\mathcal{B}_4'$ hold. Let $X \subseteq \Delta_i$ be a subset of Δ_i and let y be an individual $y \in ((X^i)_d)^j$. Since all GCIs of the form $(X^i)_d \sqsubseteq (X^i)_{d+1}$ follow from $\mathcal{B}_4'$ it holds that

$$((X^i)_d)^j \subseteq ((X^i)_{d+1})^j \subseteq ((X^i)_{d+2})^j \subseteq \dots$$

Hence $y \in ((X^i)_k)^j$ holds for all $k \geq d$. From Lemma 5.5 we know that if k is large enough then $((X^i)_k)^j = (X^i)^j$. Since $y \in ((X^i)_k)^j$ holds for arbitrarily large k it follows that $y \in (X^i)^j$ holds. We have shown that $((X^i)_d)^j \subseteq (X^i)^j$ holds in j. Hence $(X^i)_d \sqsubseteq X^i$ follows from $\mathcal{B}_4'$.

To prove completeness of $\mathcal{B}_4$ we show that every GCI from $\mathcal{B}$ follows from $\mathcal{B}_4$. Let $C \sqsubseteq D$ be a GCI from $\mathcal{B}$. We already know that $(C^{ii})_d \sqsubseteq C^{ii}$ follows from $\mathcal{B}_4'$ and thus from $\mathcal{B}_4$. By definition $\mathcal{B}_4$ contains the GCI $C_d \sqsubseteq (C^{ii})_d$. Therefore $C_d \sqsubseteq C^{ii}$ follows from $\mathcal{B}_4$. Since $C_d \sqsubseteq C$ holds for all unravellings the GCI $C \sqsubseteq C^{ii}$ also follow from $\mathcal{B}_4$. Finally, Lemma 4.3 shows that $C \sqsubseteq D$ follows from $\mathcal{B}_4$. We have shown that all GCIs from $\mathcal{B}$ follow from $\mathcal{B}_4$. Since $\mathcal{B}$ is complete for i the set of implications $\mathcal{B}_4$ must be complete as well. □

6 Exploration of $\mathcal{EL}^{\perp}_{\text{gfp}}$-Models

In the previous chapter we have presented a method for computing a finite base for an a priori given model. This base can be used as a starting point for an ontology. There is one problem with this approach. If the chosen model was to restricted it possibly satisfies GCIs that do not hold in all intended models of the knowledge base. In this case the knowledge engineer has to weaken or remove some of the GCIs. In this chapter we present a procedure to assist the knowledge engineer. Our procedure has been inspired by Attribute Exploration from FCA and is therefore called *Model Exploration*. Like Attribute Exploration, which begins with a smaller context that is extended until it contains all relevant counterexamples, Model Exploration starts with a smaller model that is extended by adding counterexamples. Upon termination, Model-Exploration produces as output the final model and a finite base for the implications holding in it.

The first section of this chapter addresses a problem that arises from the nature of the base $\mathcal{B}_3$ from the previous chapter. To compute the base $\mathcal{B}_3$ for a given model i we need to generate an induced context whose attribute set M_i is obtained from the model i itself. In the beginning of an exploration procedure the model i is not known completely and neither is M_i. We therefore need an algorithm for computing a finite base for i, where attributes are added on the fly as they become known. Such an algorithm is presented in the first section. In the second part we present an exploration formalism based on the algorithm from the first section.

6.1 A Practical Algorithm for Computing Bases

In the previous section we have shown how a finite base $\mathcal{B}_3$ for a given model i can be obtained. In this section we present an alternative algorithm that can also compute a finite base for a model i. Instead of computing the complete set of attributes M_i in advance, it starts with a smaller set of attributes M_0 which is gradually enlarged. In the first part of this section we describe the underlying FCA theory. We show that Next-Closure can be modified to deal with growing sets of attributes. The second part describes our algorithm for computing a finite base, which is based on this modification of Next-Closure.

6.1.1 A Next-Closure-Algorithm for Growing Sets of Attributes

The Procedure without Background Knowledge

Let $\mathbb{K} = (G, M, I)$ be a context. We present an algorithm (Algorithm 7) that computes an implicational base of $\mathbb{K}$. Algorithm 7 is a modification of the Next-Closure-algorithm for computing the Duquenne-Guigues Base of a given context (Algorithm 3). It is tailored to the following scenario. We want to compute an implicational base for a context $\mathbb{K}$, whose set of attributes initially is only partially known. We assume that the input of the algorithm is a context $\mathbb{K}_0 = (G, M_0, I_0)$, where $M_0 \subseteq M$ and $I_0 = I \cap (G \times M_0)$. As we find new implications more attributes become known. For now, we assume that the attributes are obtained from an unspecified outside source. Neither do we make restrictions to the nature of this outside source nor to the attributes themselves. We show that Next-Closure is robust enough to obtain an implicational base (or an $\mathcal{S}$-base if background knowledge is present) even if the set of attributes is growing during the process of computing new attributes.

Some readers may be familiar with Object Exploration [Stu96c]. Object Exploration is the dual setting to Attribute Exploration where all objects of a context are known and attributes are added by a human expert. Even though both formalisms start with an incomplete set of attributes they should not be confused. Object Exploration computes a base for implications between sets of objects, while our algorithm com-

putes implications between sets of attributes. In object exploration new attributes are required to be counterexamples to previously rejected object implications. We do not make such a requirement.

In each iteration of Algorithm 7 we compute a new left-hand side P_k for an implication. Afterwards, the outside source is allowed to add new attributes, which yields a new context $\mathbb{K}_{k+1} = (G, M_{k+1}, I_{k+1})$. The new set of attributes M_{k+1} must contain the previous set of attributes M_k and the incidence relation I_{k+1} must coincide with I_k on M_k, i. e. $M_k \subseteq M_{k+1} \subseteq M$ and $I_k = I_{k+1} \cap (G \times M_k)$. We furthermore require the new attributes to be smaller than the previously known attributes with respect to the total order $<$ on M. This ensures that all sets containing a new attribute are lectically larger than all previously computed left-hand sides.

Once the new context $\mathbb{K}_{k+1}$ has been obtained we need to take care of the fact that when the attribute set is extended the closure operator $\cdot''$ also changes. Therefore after adding new attributes, the right-hand sides of all previously computed implications need to be updated yielding a new set of implications $\mathcal{L}_{k+1}$. In order to make it clear which context we are referring to when using the derivation operators we add an index, e. g. $P_k''^k$ denotes the set that is obtained by applying the two derivation operators from $\mathbb{K}_k$ to P_k.

At the end of each iteration a new left-hand side P_{k+1} is obtained as the next closure of P_k with respect to $\mathcal{L}_{k+1}(\cdot)$, i. e. the lectically smallest set of attributes that is lectically greater than P_k and respects all implications from $\mathcal{L}_{k+1}$. It can be computed effectively using Lemma 3.2.

Provided that from a certain point on no new attributes are added it is obvious that Algorithm 7 terminates. The final set of attributes M has only finitely many subsets and the lectic order ensures that no subset of M is obtained twice as a left-hand side. Therefore, the number of iterations is bounded by $2^{|M|}$.

Lemma 6.1 (Termination of Algorithm 7)**.** *If there is a natural number $r \in \mathbb{N}$ such that $M_k = M_r$ for all $k \geq r$ then Algorithm 7 terminates.*

Assume that Algorithm 7 terminates after the n-th iteration for some $n \in \mathbb{N}$. Its output is the set of implications $\mathcal{L}_n$ which is obtained as

$$\mathcal{L}_n = \{P_r \to P_r''^n \mid P_r \neq P_r''^n, r \in \{0, \ldots, n-1\}\}.$$

Algorithm 7 Computing a Base for the Case of a Growing Set of Attributes

input $\mathbb{K}_0$
$P_0 := \emptyset$, $k := 0$ {initialization}
while $P_k \neq$ null **do**
  input $\mathbb{K}_{k+1} = (G, M_{k+1}, I_{k+1})$ {obtaining new attributes}
  $\mathcal{L}_{k+1} := \{P_r \to P_r''^{k+1} \mid P_r \neq P_r''^{k+1}, r \in \{0, \ldots, k\}\}$ {updating implications}
  if $P_k = M_k = M_{k+1}$ **then**
    $P_{k+1} :=$ null
  else
    $P_{k+1} :=$ lectically smallest subset of M_{k+1} that is
    – lectically greater than P_k, and
    – respects all implications from $\mathcal{L}_{k+1}$.
  end if
  $k := k + 1$
end while
return $\mathcal{L}_k$

We will now present an algorithm that extends Algorithm 7 by introducing background knowledge. Algorithm 7 is a special case of Algorithm 8 where we define the background knowledge to be empty. Therefore we can postpone the proof than $\mathcal{L}_n$ is a base for $\mathbb{K}_n$.

The Procedure with Background Knowledge

Algorithm 8 is an extension of Algorithm 7 that allows for background knowledge. If we decide to allow growing sets of attributes in Algorithm 5 it is natural to also allow the background knowledge to grow. When a new attribute is added there is likely some background knowledge about this new attribute that needs to be added as well. Thus, we start with an original set of background knowledge $\mathcal{S}_0$ and allow that in the k-th iteration a new set of background knowledge $\mathcal{S}_k$ can be provided (again from an unspecified source). This new set $\mathcal{S}_k$ has to contain the previous background knowledge $\mathcal{S}_{k-1}$.

Like for Algorithm 7, it is not hard to see that Algorithm 8 terminates if from a certain point on no new attributes are added.

Algorithm 8 Computing a Base for the Case of a Growing Set of Attributes with Background Knowledge

1: **input** $\mathbb{K}_0$, $\mathcal{S}_0$ {context $\mathbb{K}_0$ and background knowledge $\mathcal{S}_0$}
2: $P_0 := \emptyset$, $k := 0$ {Initialization}
3: **while** $P_k \neq \mathsf{null}$ **do**
4: **input** $\mathbb{K}_{k+1} = (G, M_{k+1}, I_{k+1})$ {obtaining new attributes}
5: **input** $\mathcal{S}_{k+1}$ {obtaining new background knowledge}
6: $\mathcal{L}_{k+1} := \{P_r \to P_r''^{k+1} \mid P_r \neq P_r''^{k+1}, r \in \{0, \ldots, k\}\}$ {updating implications}
7: **if** $P_k = M_k = M_{k+1}$ **then**
8: $P_{k+1} := \mathsf{null}$
9: **else**
10: $P_{k+1} :=$ lectically smallest subset of M_{k+1} that is
– lectically greater than P_k, and
– respects all implications from $\mathcal{L}_{k+1} \cup \mathcal{S}_{k+1}$.
11: **end if**
12: $k := k + 1$
13: **end while**
14: **return** $\mathcal{L}_k$

Lemma 6.2 (Termination of Algorithm 8)**.** *If there is a natural number $n \in \mathbb{N}$ such that $M_k = M_n$ for all $k \geq n$ then Algorithm 8 terminates.*

We now assume that Algorithm 8 has terminated after the n-th iteration for some $n \in \mathbb{N}$. We want to prove that the final set of implications $\mathcal{L}_n$ is an $\mathcal{S}_n$-base of the final context $\mathbb{K}_n$.

Lemma 6.3. *Let $Q \subseteq M_n$ be a subset of the final set of attributes M_n. Then*

- *if $Q = Q''^n$ then there is some $k \in \{0, \ldots, n\}$ such that $Q = P_k$, and*
- *if $Q \neq Q''^n$ then Q does not respect all implications from $\mathcal{L}_n \cup \mathcal{S}_n$.*

Proof. The case $Q = \emptyset$ is trivial because of $P_0 = \emptyset$. The following arguments apply to the case $Q \neq \emptyset$ independently whether $Q = Q''^n$ or $Q \neq Q''^n$ holds. Since Algorithm 8 has terminated after the n-th iteration it holds that $P_n = \mathsf{null}$ and $P_{n-1} = M_{n-1} = M_n$ since otherwise the algorithm cannot leave the while-loop. $P_{n-1} = M_n \geq Q$ holds with respect to the lectic order, because the lectic order extends the subset order and $Q \subseteq M_n$ holds. Thus there is a set P_k that is lectically greater or equal to Q. Since there are only finitely many P_k there must be some natural number $r \in \{1, \ldots, n-1\}$ such that $P_r \geq Q$ and $P_{r-1} < Q$ holds with respect to the lectic order. We show that all attributes from Q are already known in the r-th iteration, i. e. $Q \subseteq M_r$ holds. Assume that Q contains an attribute $m \in M_n \setminus M_r$. Attributes that are added later are smaller than attributes that have been added earlier. Therefore Q would be lectically greater than any subset of M_r and in particular greater than P_r. Hence such an attribute m cannot exist and therefore $Q \subseteq M_r$ must hold.

We consider the case $Q = Q''^n$. As a concept intent Q''^n respects all implications that hold in $\mathbb{K}_n$. In particular it respects all implications from $\mathcal{L}_r \cup \mathcal{S}_r$. Furthermore, Q satisfies $P_{r-1} < Q \leq P_r$. Therefore, in the r-th iteration of Algorithm 8 Q is the lectically next subset of M_r such that $P_{r-1} < Q$ and Q respects all implications from $\mathcal{L}_r \cup \mathcal{S}_r$. Hence it holds that $Q = P_r$.

We now look at the second case $Q \neq Q''^n$. If there is an implication from $\mathcal{S}_n$ that is not respected by Q then the original claim holds trivially. Furthermore, if $Q = P_r$ then $\mathcal{L}_n$ contains the implication $Q \to Q''^n$

which is not respected by Q, because we know that $Q \neq Q''^n$. Therefore the claim also holds.

We examine the remaining case where Q respects all implications from S_n and $Q \neq P_r$ holds. Assume that Q respects all implications from $\mathcal{L}_r$. Q respects all implications from $\mathcal{S}_n$ and thus also all implications from $\mathcal{S}_r \subseteq \mathcal{S}_n$. Then Q is the lectically smallest subset of M_r that satisfies $Q > P_{r-1}$ and respects all implications from $\mathcal{L}_r \cup \mathcal{S}_r$. This contradicts the fact that $Q \neq P_r$. Therefore, the assumption that Q respects all implications from $\mathcal{L}_r$ must be false.

We have thus shown that there must be an implication $P \to P''^r$ from $\mathcal{L}_r$ that is not respected by Q, i. e.

$$P \subseteq Q, \text{ and } P''^r \not\subseteq Q.$$

We show that Q does not respect $P \to P''^n$ either. Since the incidence relation I_n from $\mathbb{K}_n$ extends I_r from $\mathbb{K}_r$ it holds that

$$P'^r = \{g \in G \mid \forall m \in P \colon gI_r m\} = \{g \in G \mid \forall m \in P : gI_n m\} = P'^n.$$

And furthermore since $M_r \subseteq M_n$ it holds that

$$\begin{aligned} P''^r &= \{m \in M_r \mid \forall g \in P'^r \colon gI_r m\} \\ &\subseteq \{m \in M_n \mid \forall g \in P'^n : gI_n m\} = P''^n. \end{aligned}$$

$P''^r \subseteq P''^n$ implies $P''^n \not\subseteq Q$ and therefore Q does not respect $P \to P''^n$. □

Using Lemma 6.3 it requires only a standard argument from FCA to show that $\mathcal{L}_n \cup \mathcal{S}_n$ is complete.

Theorem 6.4. *Upon termination the set $\mathcal{L}_n$ is an $\mathcal{S}_n$-base of $\mathbb{K}_n$.*

Proof. $\mathcal{L}_n$ is sound for $\mathbb{K}_n$ since it contains only implications of the form $Q \to Q''^n$. To prove that $\mathcal{L}_n \cup \mathcal{S}_n$ is complete we show that all implications of the form $Q \to Q''^n$ follow from $\mathcal{L}_n \cup \mathcal{S}_n$. Let $Q \subseteq M_n$ be a set of attributes, and let $R \subseteq M_n$ be a set of attributes that respects all implications from $\mathcal{L}_n$. We show that $Q \to Q''^n$ follows from $\mathcal{L}_n \cup \mathcal{S}_n$ by proving that R also respects $Q \to Q''^n$. From Lemma 6.3 it follows that $R = R''^n$. If $Q \not\subseteq R$ then R trivially respects $Q \to Q''^n$. If $Q \subseteq R$ then $R = R''^n$ yields

$$Q \subseteq R''^n.$$

Lemma 3.1 yields

$$Q''^n \subseteq (R''^n)''^n$$

and therefore

$$Q''^n \subseteq R''^n = R$$

follows from Lemma 3.1. Hence R respects $Q \to Q''^n$. We have thus shown that $Q \to Q''^n$ follows from $\mathcal{L}_n \cup \mathcal{S}_n$. Since $\{Q \to Q''^n \mid Q \subseteq M_n\}$ is complete for $\mathbb{K}$ (cf. Lemma 3.4) the set $\mathcal{L}_n \cup \mathcal{S}_n$ is also complete for $\mathbb{K}$. Correctness of $\mathcal{L}_n$ follows immediately from Lemma 3.1. Thus $\mathcal{L}_n$ is an $\mathcal{S}_n$-base for $\mathbb{K}_n$. □

As mentioned previously Algorithm 7 is a special case of Algorithm 8 where we define $\mathcal{S}_k = \emptyset$ for all $k \in \{0, \ldots, n\}$. This gives us the following corollary.

Corollary 6.5. *Assume that Algorithm 7 terminates after the n-th iteration. Then the set $\mathcal{L}_n$ which is returned by Algorithm 7 is a base for $\mathbb{K}_n$.*

Algorithms 7 and 8 show that Next-Closure is robust enough to deliver a base of a context $\mathbb{K}$, even if not all attributes of that context are known from the beginning. Unfortunately, in contrast to the classical Algorithms 3 and 5 the resulting base is not guaranteed to have minimal cardinality. We illustrate this using an example.

Example 6.1. We start with the context $\mathbb{K}_0$ shown in Table 6.1 with the attribute set $M_0 = \{A\}$. No new attributes are added in the first two iterations. The first two left-hand sides that Algorithm 7 finds are $\emptyset$ and $\{A\}$. If no new attributes were added now, the algorithm would terminate. However, we assume that an attribute B is added yielding the context $\mathbb{K}_2$ from Table 6.2. The next two left-hand sides that Algorithm 7 finds are $\{B\}$ and $\{A, B\}$. A new attribute C is added in the fourth iteration yielding $\mathbb{K}_4$ from Table 6.3. After this, no more attributes are added and we obtain the last two left-hand sides which are $\{C\}$ and $\{A, B, C\}$. The final output is the base $\mathcal{L}_6$ consisting of the implications

- $\{A\} \to \{A, C\}$,
- $\{A, B\} \to \{A, B, C\}$, and

Table 6.1: $\mathbb{K}_0 = \mathbb{K}_1$

	A
a	×
b	

Table 6.2: $\mathbb{K}_2 = \mathbb{K}_3$

	A	B
a	×	
b		×

Table 6.3: $\mathbb{K}_4 = \mathbb{K}_5 = \mathbb{K}_6$

	A	B	C
a	×		×
b		×	

- $\{C\} \to \{A, C\}$.

The other left-hand sides $\emptyset$, $\{B\}$, and $\{A, B, C\}$ are closed with respect to $\cdot''^{6}$ and therefore they are not used in $\mathcal{L}_6$. The interesting implication is $\{A, B\} \to \{A, B, C\}$. Even though this is a non-trivial implication, it is still redundant, since it follows from $\{A\} \to \{B, C\}$.

Algorithms 7 and 8, like Algorithm 3, do not only compute the left-hand sides for the implications in $\mathcal{L}_n$. They also find the concept intents of $\mathbb{K}_n$. Unlike in Algorithm 3 it is not possible to decide during runtime whether a set P_k is a concept intent of $\mathbb{K}_n$. Even if $P_k = P_k^{\prime\prime k}$ holds it might be the case that $P_k \neq P_k^{\prime\prime n}$ because the attributes in $P_k^{\prime\prime n} \setminus P_k^{\prime\prime k}$ are added at a later point in time. In this case it is not possible to find out during the k-th iteration if the left-hand-side P_k is redundant, like the set $\{A, B\}$ in the example, or if it is not redundant, like the set $\{A\}$.

6.1.2 Computing M_i on the Fly

In Section 5.2.1 we have presented the base $\mathcal{B}_{\mathcal{DG}}$ for a given model i. It is obtained from the $\mathcal{S}_i$-Duquenne-Guigues Base of the context $\mathbb{K}_i$, where $\mathbb{K}_i$ is the context induced by M_i and i, and where M_i is the following set of concept descriptions

$$M_i = \{\bot\} \cup \mathcal{N}_C \cup \{\exists r.X^i \mid r \in \mathcal{N}_R \text{ and } X \subseteq \Delta_i, X \neq \emptyset\}.$$

Computationally, the two most time-consuming tasks when computing $\mathcal{B}_{\mathcal{DG}}$ are computing the set M_i (cf. Algorithm 6), and computing the Duquenne-Guigues Base of $\mathbb{K}_i$ (since no output polynomial algorithm for enumerating pseudo-intents is known). The first task of computing M_i is particularly problematic for two reasons. Firstly, it has to be performed before the actual computation of GCIs starts. Therefore it

causes a considerable delay between the start and when the first GCI is obtained. Secondly, as mentioned before, in an exploration setting where i is initially unknown it is not possible to compute M_i beforehand.

Instead of computing the complete set of attributes in the beginning we propose to start with the set of attributes

$$M_0 = \mathcal{N}_C \cup \{\bot\}$$

and add new attributes on the fly. We basically apply Algorithm 8 to the context that is induced by M_0 and i. Whenever Algorithm 8 generates a new left-hand side P we compute $(\bigsqcap P)^{ii}$ and add the $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions of the form $\exists r.(\bigsqcap P)^{ii}$ for all role names $r \in \mathcal{N}_R$ to the set of attributes. More precisely, we add new attributes only if no equivalent attribute is already present. We denote this by $\dot\cup$ instead of $\cup$. Whenever new attributes have been added the background knowledge is updated according to (5.16). Algorithm 9 shows an instance of Algorithm 8 obtained this way.

We know that Algorithm 8 terminates if from a certain point on no new attributes are added. Algorithm 9 is basically Algorithm 8 applied to the induced context $\mathbb{K}_i$. The attributes that are added in Line 6 all come from the set M_i which is finite. Therefore, Algorithm 9 must terminate.

Lemma 6.6. *Algorithm 9 terminates for every finite input model* $i = (\Delta_i, \cdot^i)$.

Assume that Algorithm 9 terminates after the n-th iteration. We know from Theorem 6.4 that $\mathcal{L}_n$ is an $\mathcal{S}_n$-base of $\mathbb{K}_n$. In this section we show that

$$\mathcal{B}_5 = \{\bigsqcap P \to (\bigsqcap P)^{ii} \mid P \to P''^n \in \mathcal{L}_n\}$$

is a base for the GCIs holding in i. $\mathbb{K}_n$ is the context that is induced by M_n and i. By Corollary 5.14 it it suffices to show that $M_n \doteq M_i$ in order to proof that $\mathcal{B}_5$ is a base for the GCIs holding in i.

Lemma 6.7. *Let* $U \subseteq M_n$ *be a subset of the final set of attributes* M_n. *Then for every role name* $r \in \mathcal{N}_R$ *there is an attribute* $C \in M_n$ *such that*

$$C \equiv \exists r.(\bigsqcap U)^{ii}.$$

Algorithm 9 Computing a Base for the GCIs Holding in an A Priori Given Model

1: **input** $i = (\Delta_i, \cdot^i)$ {model i}
2: $P_0 := \emptyset$, $M_0 := \mathcal{N}_C \cup \{\bot\}$, $k := 0$ {initialization}
3: $\mathbb{K}_0 :=$ the context induced by M_0 and i
4: $\mathcal{S}_0 := \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_C\}$, $\mathcal{L}_0 := \emptyset$
5: **while** $P_k \neq \mathsf{null}$ **do**
6: $M_{k+1} := M_k \,\dot\cup\, \{\exists r.(\bigsqcap P_k)^{ii} \mid r \in \mathcal{N}_R\}$ {obtaining new attributes}
7: $\mathbb{K}_{k+1} :=$ the context induced by M_{k+1} and i {updating $\mathbb{K}_{k+1}$, $\mathcal{S}_{k+1}$ and $\mathcal{L}_{k+1}$}
8: $\mathcal{L}_{k+1} := \{P_r \to P_r''^{k+1} \mid r \in \{0, \ldots, k\}\}$
9: $\mathcal{S}_{k+1} := \{\{A\} \to \{B\} \mid A, B \in M_{k+1}, A \sqsubseteq B\}$
10: **if** $P_k = M_k = M_{k+1}$ **then**
11: $P_{k+1} := \mathsf{null}$
12: **else**
13: $P_{k+1} :=$ lectically smallest subset of M_{k+1} that is
– lectically greater than P_k, and
– respects all implications from $\mathcal{L}_{k+1} \cup \mathcal{S}_{k+1}$.
14: **end if**
15: $k := k + 1$
16: **end while**
17: **return** $\{\bigsqcap P \to (\bigsqcap P)^{ii} \mid P \to P''^k \in \mathcal{L}_k\}$

Proof. From Lemma 4.10 and Lemma 4.1 we obtain

$$(\bigsqcap U''^n)^i = U'''^n = U'^n = (\bigsqcap U)^i.$$

Hence it holds that

$$\exists r.(\bigsqcap U''^n)^{ii} \equiv \exists r.(\bigsqcap U)^{ii}. \tag{6.1}$$

Algorithm 9 is a special case of Algorithm 8. Therefore Lemma 6.3 holds for Algorithm 9, too. It yields that there is some $k \in \{0, \ldots, n-1\}$ such that $U''^n = P_k$. Thus in Line 6 of the k+1-th iteration the $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description

$$\exists r.(\bigsqcap P_k)^{ii} = \exists r.(\bigsqcap U''^n)^{ii}$$

is added to the set of attributes. Thus M_n contains $\exists r.(\bigsqcap U''^n)^{ii}$ which is equivalent to $\exists r.(\bigsqcap U)^{ii}$ because of (6.1). □

Lemma 6.8. *It holds that* $M_n \doteq M_i$.

Proof. $M_n \dot{\subseteq} M_i$ holds since every $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description of M_n is either from $\{\bot\} \cup \mathcal{N}_C$ or it is of the form $\exists r.(\bigsqcap P)^{ii}$ for some $P \subseteq M_n$.

It remains to show that for every $X \subseteq \Delta_i$ and every $r \in \mathcal{N}_R$ there is a concept description $C \in M_n$ such that $\exists r.X^i \equiv C$. Lemma 5.6 shows that for every $X \subseteq \Delta_i$ there is an acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description D satisfying $D^i = (X^i)^i$. Applying the interpretation function on both sides yields $D^{ii} = X^{iii} = X^i$. Thus it suffices to show that for every acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description D there is a concept description $C \in M_n$ satisfying $C \equiv \exists r.D^{ii}$. We prove this by induction over the role depth of D. *Base Case:* The case where $D = \bot$ is trivial. Let $D = \bigsqcap S$ for some set $S \subseteq \mathcal{N}_C$. Then in particular $S \subseteq M_0 \subseteq M_n$ holds. Let $r \in \mathcal{N}_R$ be a role name. By Lemma 6.7 there is some concept description $C \in M_n$ such that $C \equiv \exists r.(\bigsqcap S)^{ii}$.

Step Case: Assume that $\exists r.E^{ii} \doteq M_n$ holds for all acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions E of role depth less than d and for all role names $r \in \mathcal{N}_R$. Let D be an acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description of role depth d and $s \in \mathcal{N}_R$ a role name. There is a set of concept names $U \subseteq \mathcal{N}_C$, and a set Π containing pairs of role names and $\mathcal{EL}_{\text{gfp}}$-concept descriptions such that

$$D = \bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E.$$

Then Lemma 4.2 proves

$$D^{ii} = \big(\bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E\big)^{ii} = \big(\bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E^{ii}\big)^{ii}.$$

By induction hypothesis it holds that $\exists r.E^{ii} \dot{\in} M_n$ for all pairs $(r, E) \in \Pi$. Furthermore it holds that $U \subseteq \mathcal{N}_C \subseteq M_n$. Lemma 6.7 proves that there is some concept description $C \in M_n$ satisfying

$$C \equiv \exists s.\big(\bigsqcap U \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E^{ii}\big)^{ii} \equiv \exists s.D^{ii}.$$

This proves that for every acyclic $\mathcal{EL}_{\text{gfp}}$-concept description D and every role name $s \in \mathcal{N}_R$ there is a concept description $C \in M_n$ satisfying $C \equiv \exists s.D^{ii}$. Because we can find some acyclic $\mathcal{EL}_{\text{gfp}}$-concept description D with $D^{ii} \equiv X^i$ for every $X \subseteq \Delta_i$, it follows that

$$\{\exists s.X^i \mid X \subseteq \Delta_i, X \neq \emptyset\} \dot{\subseteq} M_n.$$

$M_i \dot{\subseteq} M_n$ follows from this and $\{\bot\} \cup \mathcal{N}_C \subseteq M_n$. Together with $M_n \dot{\subseteq} M_i$ we obtain $M_i \doteq M_n$. □

Completeness of $\mathcal{B}_5$ is an immediate consequence of the fact that $\mathcal{L}_n$ is a base of $\mathbb{K}_n$, the fact that $M_n \doteq M_i$ and Corollary 5.14.

Theorem 6.9. *$\mathcal{B}_5$ is a base for the GCIs holding in i.*

Because of the effects that have been demonstrated in Example 6.1 it is possible that $\mathcal{B}_5$ is not irredundant. This shortcoming of $\mathcal{B}_5$ is counterbalanced by the fact that it is not necessary to compute the full attribute set in advance when using Algorithm 9.

6.1.3 Acyclic Left-Hand Sides

The base that we obtain from Algorithm 7 uses cyclic left-hand sides. Of course, we can apply the procedure from Section 5.3 to remove cyclic concept descriptions *after* the algorithm has terminated. However, since the semantics of cyclic concept descriptions are not always intuitive, it is desireable to avoid them already during runtime. This becomes particularly relevant in an exploration setting where GCIs need to be

confirmed by a human expert. We do not have a method to avoid cyclic concept descriptions during runtime completely, but it is fairly straightforward to avoid cyclic left-hand sides.

The reason why there can be cyclic left-hand-sides in $\mathcal{B}_5$ is that there can be cyclic $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept descriptions in the attribute set. These cyclic concept descriptions are obtained in Line 6 because the most-specific concept $(\bigsqcap P_k)^{ii}$ can be cyclic. We know from Lemma 4.2 that $(\exists r.(\bigsqcap P_k)^{ii})^i = (\exists r.(\bigsqcap P_k))^i$ holds. Therefore if we replace the attribute $\exists r.(\bigsqcap P_k)^{ii}$ by the attribute $\exists r.(\bigsqcap P_k)$ the induced context $\mathbb{K}_k$ will not change except for renaming of an attribute. Algorithm 10 is based on this observation. It differs from Algorithm 9 only in Line 6.

Algorithm 10 Computing a Base for the GCIs Holding in an A Priori Given Model Using Only Acyclic Left-Hand-Sides

1: **input** $i = (\Delta_i, \cdot^i)$ {model i}
2: $\bar{P}_0 := \emptyset$, $\bar{M}_0 := \mathcal{N}_P \cup \{\bot\}$, $k := 0$ {initialization}
3: $\bar{\mathbb{K}}_0 :=$ the context induced by $\bar{M}_0$ and i
4: $\bar{\mathcal{S}}_0 := \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_P\}$, $\bar{\mathcal{L}}_0 := \emptyset$
5: **while** $\bar{P}_k \neq$ null **do**
6: $\quad \bar{M}_{k+1} := \bar{M}_k \cup \begin{cases} \{\exists r. \bigsqcap \bar{P}_k \mid r \in \mathcal{N}_R\} & \text{if } (\bigsqcap \bar{P}_k)^{ii} \not\equiv (\bigsqcap \bar{P}_\ell)^{ii} \\ & \text{for all } \ell < k \\ \emptyset & \text{otherwise} \end{cases}$
7: $\quad \bar{\mathbb{K}}_{k+1} :=$ the context induced by $\bar{M}_{k+1}$ and i {updating $\bar{\mathbb{K}}_{k+1}$, $\bar{\mathcal{S}}_{k+1}$ and $\bar{\mathcal{L}}_{k+1}$}
8: $\quad \bar{\mathcal{L}}_{k+1} := \{\bar{P}_r \to \bar{P}_r''^{k+1} \mid r \in \{0, \ldots, k\}\}$
9: $\quad \bar{\mathcal{S}}_{k+1} := \{\{A\} \to \{B\} \mid A, B \in \bar{M}_{k+1}, A \sqsubseteq B\}$
10: $\quad$ **if** $\bar{P}_k = \bar{M}_k = \bar{M}_{k+1}$ **then**
11: $\quad\quad \bar{P}_{k+1} :=$ null
12: $\quad$ **else**
13: $\quad\quad \bar{P}_{k+1} :=$ lectically smallest subset of $\bar{M}_{k+1}$ that is
– lectically greater than $\bar{P}_k$, and
– respects all implications from $\bar{\mathcal{L}}_{k+1} \cup \bar{\mathcal{S}}_{k+1}$.
14: $\quad$ **end if**
15: $\quad k := k + 1$
16: **end while**
17: **return** $\{\bigsqcap \bar{P} \to (\bigsqcap \bar{P})^{ii} \mid \bar{P} \to \bar{P}''^{k} \in \bar{\mathcal{L}}_k\}$

We prove that Algorithm 10 terminates and that upon termination the set

$$\bar{\mathcal{B}}_5 = \{\bigsqcap \bar{P} \to (\bigsqcap \bar{P})^{ii} \mid \bar{P} \to \bar{P}'^n \in \bar{\mathcal{L}}_n\}$$

is a base for the GCIs holding in i. To this purpose we compare Algorithm 9 and Algorithm 10. Assume that Algorithm 9 and Algorithm 10 are initialized with the same model i as input. For convenience we introduce the notation m_{kr} to denote

$$m_{kr} = \exists r.(\bigsqcap P_k)^{ii}$$

the attribute added in Line 6 in the k-th iteration of Algorithm 9 for each role name $r \in \mathcal{N}_R$. Analogously,

$$\bar{m}_{kr} = \exists r.\bigsqcap \bar{P}_k$$

denotes the attribute added in Line 6 in the k-th iteration of Algorithm 10 for each role name $r \in \mathcal{N}_R$. The set of concept names M_k consists of the bottom concept $\bot$, the concept names from $\mathcal{N}_C$ and attributes of the form m_{kr}. Therefore every set P_k can be written as

$$P_k = N_k \cup \{m_{k'r'} \mid (k', r') \in \Pi_k\}$$

for some set $N_k \subseteq \{\bot\} \cup \mathcal{N}_C$ and some set $\Pi_k \subseteq \{1, \ldots, k\} \times \mathcal{N}_R$.

Lemma 6.10. *Assume that Algorithm 9 and Algorithm 10 have completed k iterations of the respective while-loops. Then $P_k = N_k \cup \{m_{k'r'} \mid (k', r') \in \Pi_k\}$, for some set $N_k \subseteq \{\bot\} \cup \mathcal{N}_C$ and some set $\Pi_k \subseteq \{1, \ldots, k\} \times \mathcal{N}_R$ implies*

$$\bar{P}_k = N_k \cup \{\bar{m}_{k'r'} \mid (k', r') \in \Pi_k\}. \tag{6.2}$$

This means that $\bar{P}_k$ is obtained from P_k simply by replacing every attribute $m_{k'r'}$ by the corresponding attribute $\bar{m}_{kr}$. Furthermore, $\bar{M}_k$ is obtained from M_k by the same kind of renaming, and

$$(\bigsqcap P_k)^{ii} \equiv (\bigsqcap \bar{P}_k)^{ii} \tag{6.3}$$

holds. For all $r \in \mathcal{N}_R$ it holds that

$$m_{kr} \sqsubseteq \bar{m}_{kr} \tag{6.4}$$

and

$$(m_{kr})^i = (\bar{m}_{kr})^i. \tag{6.5}$$

Proof. We prove this statement by induction over k. The base case $k = 0$ is obvious since $P_0 = \bar{P}_0 = \emptyset$.

Step Case: We assume that statements (6.2), (6.3), (6.4), and (6.5) hold for all $k' \in \{1, \dots, k\}$ and show that they also hold for $k+1$. In Algorithm 9 the new set of attributes M_{k+1} is obtained as $M_{k+1} = M_k \dot{\cup} \{m_{kr} \mid r \in \mathcal{N}_R\}$. New attributes m_{kr} are added iff there are no equivalent attributes in M_k. A previously added attribute $m_{k'r} = \exists r.(\bigsqcap P_{k'})^{ii}$, where $k' < k$, satisfies $m_{k'r} = \exists r.(\bigsqcap P_{k'})^{ii} \equiv \exists r.(\bigsqcap P_k)^{ii} = m_{kr}$ iff $(\bigsqcap P_{k'})^{ii} \equiv (\bigsqcap P_k)^{ii}$ holds. By induction hypothesis (6.3) holds for k and k'. Therefore, $(\bigsqcap P_{k'})^{ii} \equiv (\bigsqcap P_k)^{ii}$ holds iff $(\bigsqcap \bar{P}_{k'})^{ii} \equiv (\bigsqcap \bar{P}_k)^{ii}$. Hence, a new attribute m_{kr} is added in Algorithm 9 iff a new attribute $\bar{m}_{kr}$ is added in Algorithm 10. This proves that $\bar{M}_{k+1}$ can be obtained from M_{k+1} by renaming attributes.

We know from the induction hypothesis that $(m_{k'r'})^i = (\bar{m}_{k'r'})^i$ holds for all $k' \in \{1, \dots, k\}$. Therefore, the induced contexts $\mathbb{K}_{k+1}$ and $\bar{\mathbb{K}}_{k+1}$ are equal except for renaming attributes (Informally, we can say that all crosses are in the same places in the two contexts, only the columns have different labels). $\bar{\mathcal{L}}_{k+1}$, $\bar{\mathcal{S}}_{k+1}$ and $\bar{P}_{k+1}$ only depend on $\bar{M}_{k+1}$, $\mathbb{K}_{k+1}$ and the sets $\bar{P}_{k'}$, where $k' \leq k$. It follows that $\bar{\mathcal{L}}_{k+1}$, $\bar{\mathcal{S}}_{k+1}$ and $\bar{P}_{k+1}$ can be obtained from $\mathcal{L}_{k+1}$, $\mathcal{S}_{k+1}$ and P_{k+1}, respectively, by renaming attributes. In particular this proves (6.2) for $k+1$.

P_{k+1} can be written as $P_{k+1} = \mathcal{N}_{k+1} \cup \{m_{k'r'} \mid (k', r') \in \Pi_{k+1}\}$, for some set $\mathcal{N}_{k+1} \subseteq \{\bot\} \cup \mathcal{N}_C$ and some set $\Pi_{k+1} \subseteq \{1, \dots, k+1\} \times \mathcal{N}_R$. Then (6.2) implies $\bar{P}_{k+1} = \mathcal{N}_{k+1} \cup \{\bar{m}_{k'r'} \mid (k', r') \in \Pi_{k+1}\}$. We obtain from (6.5) and (6.2) that

$$\begin{aligned}
(\bigsqcap P_{k+1})^i &\equiv \bigcap_{A \in \mathcal{N}_{k+1}} A^i \cap \bigcap_{(k',r') \in \Pi_{k+1}} m^i_{k'r'} \\
&\equiv \bigcap_{A \in \mathcal{N}_{k+1}} A^i \cap \bigcap_{(k',r') \in \Pi_{k+1}} \bar{m}^i_{k'r'} \\
&\equiv (\bigsqcap \bar{P}_{k+1})^i
\end{aligned}$$

and thus $(\bigsqcap P_{k+1})^{ii} \equiv (\bigsqcap \bar{P}_{k+1})^{ii}$ holds, which proves (6.3) for $k+1$. This also yields $(\bigsqcap P_{k+1})^{ii} \equiv (\bigsqcap \bar{P}_{k+1})^{ii} \sqsubseteq \bigsqcap \bar{P}_{k+1}$ which implies

$$m_{k+1,r} = \exists r.(\bigsqcap P_{k+1})^{ii} \sqsubseteq \exists r. \bigsqcap \bar{P}_{k+1} = \bar{m}_{k+1,r},$$

i. e. (6.4) holds for $k+1$. Finally, from (6.3) and Lemma 4.2 we obtain

$$\begin{aligned} m^i_{k+1,r} &= \left(\exists r.\left(\bigsqcap P_{k+1}\right)^{ii}\right)^i \\ &= \left(\exists r.\left(\bigsqcap \bar{P}_{k+1}\right)^{ii}\right)^i \\ &= \left(\exists r.\left(\bigsqcap \bar{P}_{k+1}\right)\right)^i \\ &= \bar{m}^i_{k+1,r}, \end{aligned}$$

which proves (6.4) for $k+1$. □

Theorem 6.11. *Algorithm 10 terminates for every input model i after a finite number of iterations n. Upon termination the set*

$$\bar{\mathcal{B}}_5 = \{\bigsqcap P \to (\bigsqcap P)^{ii} \mid P \to P''^n \in \bar{\mathcal{L}}_n\}$$

is a base for the GCIs holding in i.

Proof. From Lemma 6.10 we obtain that $P_k = M_k = M_{k+1}$ after the k-th iteration of Algorithm 9 iff $\bar{P}_k = \bar{M}_k = \bar{M}_{k+1}$ holds after the k-th iteration of Algorithm 10 with the same input i. Thus Algorithm 10 terminates after the $k+1$-th iteration iff Algorithm 9 terminates after the $k+1$-th iteration. Since Algorithm 9 terminates this proves that Algorithm 10 also terminates on every input.

$\bar{\mathcal{B}}_5$ is a subset of $\mathcal{B}_0$, hence it must be sound. To prove completeness we show that every GCI from $\bar{\mathcal{B}}_5$ follows from the corresponding GCI in $\bar{\mathcal{B}}_5$, i. e. that $\bigsqcap P_k \to (\bigsqcap P_k)^{ii}$ follows from $\bigsqcap \bar{P}_k \to (\bigsqcap \bar{P}_k)^{ii}$ for all $k \in \{1, \ldots, n\}$. Assume that P_k can be written as

$$P_k = N_k \cup \{m_{k'r'} \mid (k', r') \in \Pi_k\}$$

for some set $N_k \subseteq \{\bot\} \cup \mathcal{N}_C$ and some set $\Pi_k \subseteq \{1, \ldots, k\} \times \mathcal{N}_R$. Lemma 6.10 yields

$$\bar{P}_k = N \cup \{\bar{m}_{k'r'} \mid (k', r') \in \Pi\}.$$

From (6.4) we obtain

$$\begin{aligned} \bigsqcap P_k &= \bigsqcap N \sqcap \bigsqcap \{m_{k'r'} \mid (k', r') \in \Pi\} \\ &\sqsubseteq \bigsqcap N \sqcap \bigsqcap \{\bar{m}_{k+1'r'} \mid (k', r') \in \Pi\} \\ &= \bigsqcap \bar{P}_k \end{aligned}$$

Also, we obtain $(\bigcap P_k)^i = (\bigcap \bar{P}_k)^i$ from (6.3). Thus we have shown $\bigcap P_k \sqsubseteq \bigcap \bar{P}_k$ and $(\bigcap P_k)^i = (\bigcap \bar{P}_k)^i$ and therefore $\bigcap P_k \sqsubseteq (\bigcap P_k)^{ii}$ follows from $\bigcap \bar{P}_k \sqsubseteq (\bigcap \bar{P}_k)^{ii}$. Since $\mathcal{B}_5$ is complete, this proves that $\bar{\mathcal{B}}_5$ is also complete. □

6.2 Model Exploration

In this section we extend Algorithm 9 in order to obtain a knowledge exploration algorithm which we call *Model Exploration*. Like Attribute Exploration from FCA our algorithm can capture knowledge by querying an expert. This expert need not necessarily be human. The expert is assumed to have complete knowledge about the domain. We require that this knowledge can be represented in the form of a finite model $i = (\Delta_i, \cdot^i)$. This model i is called the *background model* of the exploration. Initially, i is unknown, or only partially known to the algorithm. The partial model i_0 that is known to the algorithm is called *working model*. The exploration formalism's goal is to compute a base for the GCIs holding in the background model i. In each iteration a new GCI is presented to the expert who can either refute or accept it. If it is refuted the expert is asked to present a counterexample. There are different ways to represent counterexamples. In this chapter we look at the situation where counterexamples are represented in the working models, i. e. using a closed-world semantics, while the following chapter deals with counterexamples that are represented in an ABox, i. e. using open-world semantics.

While we want the working models to contain counterexamples to the GCIs that have been refuted, they should obviously not contain counterexamples to GCIs that do hold in the background model i. To ensure that this is the case we require all working models to be *connected submodels* of i, where we define submodels and connected submodels as follows.

Definition 6.1. The model $j = (\Delta_j, \cdot^j)$ is called a *submodel* of i if

- $\Delta_j \subseteq \Delta_i$,
- $\text{names}_j(x) = \text{names}_i(x)$ for all $x \in \Delta_j$
- $\text{succ}^r_j(x) \subseteq \text{succ}^r_i(x)$ for all $x \in \Delta_j$ and all $r \in \mathcal{N}_R$.

The model j is called a *connected submodel* of i if it is a submodel of i and additionally $\mathrm{succ}_i^r(x) \subseteq \Delta_j$ and $\mathrm{succ}_j^r(x) = \mathrm{succ}_i^r(x)$ holds for all $x \in \Delta_j$ and all $r \in \mathcal{N}_R$. If j is a submodel of i we say that i *extends* j.

In order to add a counterexample for a GCI $C \sqsubseteq D$ the expert must provide a new working model i_l that extends the previous working model i_{l-1}. The new model i_l must contain a counterexample, i. e. $C^{i_l} \not\subseteq D^{i_l}$ must hold. We further require that i_l is a connected submodel of the working model i. In Example 6.2 we shall see that it is necessary to require connectedness in order to ensure that i_l does not contain a counterexample to a GCI that does hold in i. The corollary to the following lemma shows that it is also sufficient.

Lemma 6.12. *Let $j = (\Delta_j, \cdot^j)$ be a connected submodel of $i = (\Delta_i, \cdot^i)$. Then for every $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-concept description $C = (A_C, \mathcal{T}_C)$ it holds that $C^j = C^i \cap \Delta_j$.*

Proof. We need to prove $C^j \subseteq C^i \cap \Delta_j$ and $C^i \cap \Delta_j \subseteq C^j$. We give a detailed proof for the second inclusion since it is more interesting.

Let $x \in \Delta_j$ be an individual satisfying $x \in C^i$. By Lemma 2.2 there is a simulation Z from C to x in i. We show that $\bar{Z} = Z \cap (\mathcal{N}_D(\mathcal{T}_C) \times \Delta_j)$ is a simulation from C to x in j. *(S1′)* Let $(B, y) \in \bar{Z}$ be a pair in the relation $\bar{Z}$. Since $\bar{Z} \subseteq Z$ and Z is a simulation from C to x in i the pair (B, y) satisfies $\mathrm{names}_{\mathcal{T}_C}(B) \subseteq \mathrm{names}_i(y)$. Since j is a submodel of i we obtain that $\mathrm{names}_i(y) = \mathrm{names}_j(y)$ holds for all $y \in \Delta_j$. Thus (B, y) also satisfies $\mathrm{names}_{\mathcal{T}_C}(B) \subseteq \mathrm{names}_j(y)$.

(S2′) Let $(B, y) \in \bar{Z}$ be a pair in the relation $\bar{Z}$, let $r \in \mathcal{N}_R$ be a role name and $B' \in \mathrm{succ}^r_{\mathcal{T}_C}(B)$ an r-successor of B. Since $(B, y) \in \bar{Z} \subseteq Z$ and Z is a simulation there must be some $y' \in \mathrm{succ}_i^r(y)$ satisfying $(B', y') \in Z$. Because j is a connected submodel of i we obtain $y' \in \Delta_j$ and $\mathrm{succ}_i^r(y) = \mathrm{succ}_j^r(y)$. Therefore $y' \in \Delta_j$ satisfies $y' \in \mathrm{succ}_j^r(y)$ and $(B', y') \in Z'$. *(S3′)* It holds that $x \in \Delta_j$ and $(A_C, x) \in Z$. Hence it also holds that $(A_C, x) \in \bar{Z}$.

We have thus shown that $\bar{Z}$ is a simulation from C to x in j. It follows from Lemma 2.2 that $x \in C^j$ holds.

To prove the second inclusion $C^j \subseteq C^i \cap \Delta_j$ let $x \in C^j$ be an individual. From Lemma 2.2 we obtain that there is a simulation Z' from C to x in j. One can readily verify that Z' is also a simulation from C to x in i which yields $x \in C^i$. □

Algorithm 11 Exploration Algorithm for Models

input $i_0 = (\Delta_{i_0}, \cdot^{i_0})$
$\bar{P}_0 := \emptyset$, $\bar{M}_0 := \mathcal{N}_C \cup \{\bot\}$, $k := 0$, $l := 0$ {Initialization}
$\bar{\mathbb{K}}_0 :=$ the context induced by $\bar{M}_0$ and i_0, $\bar{\mathcal{S}}_0 := \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_C\}$, $\bar{\mathcal{L}}_0 := \emptyset$
while $\bar{P}_k \neq$ null **do**
  while Expert refutes $\bigsqcap \bar{P}_k \sqsubseteq (\bigsqcap \bar{P}_k)^{i_l i_l}$ **do**
    Ask the expert for a new working model i_{l+1} that
    – extends i_l,
    – is a connected submodel of the background model i, and
    – contains a counterexample for $\bigsqcap \bar{P}_k \sqsubseteq (\bigsqcap \bar{P}_k)^{i_l i_l}$
    $l := l + 1$
  end while
  $\bar{M}_{k+1} := \bar{M}_k \cup \{\exists r.(\bigsqcap \bar{P}_k)^{i_l i_l} \mid r \in \mathcal{N}_R\}$ {obtaining new attributes}
  $\bar{\mathbb{K}}_{k+1} :=$ the context induced by $\bar{M}_{k+1}$ and i_l
  $\bar{\mathcal{L}}_{k+1} := \{\bar{P}_r \to \bar{P}_r''^{k+1} \mid r \in \{0, \ldots, k\}, \bar{P}_r \neq \bar{P}_r''^{k+1}\}$
  $\bar{\mathcal{S}}_{k+1} := \{\{A\} \to \{B\} \mid A, B \in \bar{M}_{k+1}, A \sqsubseteq B\}$
  if $\bar{P}_k = \bar{M}_k = \bar{M}_{k+1}$ **then**
    $\bar{P}_{k+1} :=$ null
  else
    $\bar{P}_{k+1} :=$ lectically smallest subset of $\bar{M}_{k+1}$ that is
    – lectically greater than $\bar{P}_k$, and
    – respects all implications from $\bar{\mathcal{L}}_{k+1} \cup \bar{\mathcal{S}}_{k+1}$.
  end if
  $k := k + 1$
end while
return $\bar{\mathcal{L}}_k$

Corollary 6.13. *Let $j = (\Delta_j, \cdot^j)$ be a connected submodel of $i = (\Delta_i, \cdot^i)$. Let C and D be $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. If $C \sqsubseteq D$ holds in i then $C \sqsubseteq D$ also holds in j.*

Proof. If $C \sqsubseteq D$ holds in i then by definition $C^i \subseteq D^i$ holds. Using Lemma 6.12 we obtain

$$C^j = C^i \cap \Delta_j \subseteq D^i \cap \Delta_j = D^j.$$

Hence, $C \sqsubseteq D$ holds in j. □

In Algorithm 9 new attributes of the form $\exists r.(\bigsqcap P_k)^{ii}$ are added in each iteration. In Model Exploration only a submodel i_l of i is known in the k-th iteration, which means that we cannot compute $(\bigsqcap P_k)^{ii}$ but only $(\bigsqcap P_k)^{i_l i_l}$. Luckily, there is an elegant way to obtain the information whether $(\bigsqcap P_k)^{i_l i_l} \equiv (\bigsqcap P_k)^{ii}$ holds. We ask the expert whether the GCI $\bigsqcap P_k \sqsubseteq (\bigsqcap P_k)^{i_l i_l}$ holds in the background model i. The following result shows that $(\bigsqcap P_k)^{i_l i_l} \equiv (\bigsqcap P_k)^{ii}$ holds if and only if the expert accepts.

Lemma 6.14. *Let $j = (\Delta_j, \cdot^j)$ be a connected submodel of $i = (\Delta_i, \cdot^i)$. If $C \sqsubseteq C^{jj}$ holds in i then j satisfies $C^{jj} \equiv C^{ii}$.*

Proof. $C \sqsubseteq C^{jj}$ holds in i and thus $C^i \subseteq (C^{jj})^i$ holds. Lemma 4.1 (7) implies

$$C^{ii} \sqsubseteq C^{jj}.$$

On the other hand $C \sqsubseteq C^{ii}$ holds in i. Corollary 6.13 implies that $C \sqsubseteq C^{ii}$ also holds in j. Thus $C^j \subseteq (C^{ii})^j$ holds and Lemma 4.1 (7) implies

$$C^{jj} \sqsubseteq C^{ii}.$$

Thus C^{jj} and C^{ii} are equivalent. □

Lemma 6.14 shows that we can obtain the correct right-hand side for a GCI by querying the expert. This leads us to our Model Exploration formalism, Algorithm 11. The only difference between Algorithm 11 and Algorithm 9 is the inner while-loop, Lines 5 to 8, where all expert interaction takes place.

Since the set M_i is finite, only finitely many attributes can be added in Algorithm 11. This means that the outer while-loop can only be entered

a finite number of times. With every pass of the inner while-loop, the working model is extended. Since the working models are submodels of the finite background model, this can only happen a finite number of times. This shows that Algorithm 11 terminates after a finite number of steps.

Lemma 6.15. *Algorithm 11 terminates for every finite background model i.*

Theorem 6.16. *Assume that Algorithm 11 terminates after the n-th iteration of the outer while-loop and that i_ℓ is the final working model. Then*

$$\mathcal{B}_6 = \{\bigsqcap \bar{P} \to (\bigsqcap \bar{P})^{i_\ell i_\ell} \mid \bar{P} \to \bar{P}''^{n} \in \bar{\mathcal{L}}_n\}$$

is a finite base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in i.

Proof. We prove that Algorithm 11 with the working model i_0 as input has the same output as Algorithm 9 with the full background model i as input. To do this we show that $\bar{P}_k \doteq P_k$, $\bar{M}_k \doteq M_k$, $\bar{\mathcal{L}}_k \doteq \mathcal{L}_k$ and $\bar{\mathcal{S}}_k \doteq \mathcal{S}_k$ holds for all $k \in \{0, \dots, n\}$, where for sets of implications we define $\doteq$ as follows. We call two implications *equal up to equivalence* if both their left-hand sides and their right-hand sides are equal up to equivalence, respectively. Two sets of implications $\mathcal{S}_1$ and $\mathcal{S}_2$ are called *equal up to equivalence* if for each implication in $\mathcal{S}_1$ there is an implication in $\mathcal{S}_2$ that is equal up to equivalence, and vice versa.

We use induction over k to prove that $\bar{P}_k \doteq P_k$, $\bar{M}_k \doteq M_k$, $\bar{\mathcal{L}}_k \doteq \mathcal{L}_k$ and $\bar{\mathcal{S}}_k \doteq \mathcal{S}_k$ hold for all $k \in \{0, \dots, n\}$. *Base Case:* For $k = 0$ in both algorithms the sets $\bar{P}_0$ and P_0 obtain the initial value $\bar{P}_0 = P_0 = \emptyset$. Likewise, $\bar{M}_0 = M_0 = \mathcal{N}_C \cup \{\bot\}$, $\bar{\mathcal{S}}_0 = \mathcal{S}_0 = \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_C\}$, and $\bar{\mathcal{L}}_0 = \mathcal{L}_0 = \emptyset$ hold.

Step Case: Assume that $\bar{P}_m \doteq P_m$, $\bar{M}_m \doteq M_m$, $\bar{\mathcal{S}}_m \doteq \mathcal{S}_m$, as well as $\bar{\mathcal{L}}_m \doteq \mathcal{L}_m$ holds for all $m \leq k$. Algorithm 11 can only reach Line 9 if the expert has confirmed that $\bigsqcap \bar{P}_k \sqsubseteq (\bigsqcap \bar{P}_k)^{i_l i_l}$ holds in the background model i. Lemma 6.14 shows that $(\bigsqcap \bar{P}_k)^{i_l i_l} \equiv (\bigsqcap \bar{P}_k)^{ii}$ holds and thus by induction hypothesis $(\bigsqcap \bar{P}_k)^{i_l i_l} \equiv (\bigsqcap P_k)^{ii}$ holds. Likewise, $(\bigsqcap \bar{P}_m)^{i_l i_l} \equiv (\bigsqcap P_m)^{ii}$ holds for all $m < k$ since $\bigsqcap \bar{P}_m \sqsubseteq (\bigsqcap \bar{P}_m)^{i_s i_s}$ has been confirmed in earlier iterations for some working model i_s that is a

submodel of i_l. This proves

$$\begin{aligned}\bar{M}_{k+1} &= \bar{M}_k \cap \{\exists r.(\bigsqcap \bar{P}_k)^{i_l i_l} \mid r \in \mathcal{N}_R\} \\ &\doteq M_k \cap \{\exists r.(\bigsqcap P_k)^{ii} \mid r \in \mathcal{N}_R\} \\ &= M_{k+1}.\end{aligned}$$

We readily obtain $\bar{\mathcal{S}}_{k+1} \doteq \mathcal{S}_{k+1}$ since both $\bar{\mathcal{S}}_{k+1}$ and $\mathcal{S}_{k+1}$ only depend on $\bar{M}_{k+1}$ and M_{k+1}, respectively. We obtain from Corollary 4.13 that

$$\bar{P}''^k_m = \mathrm{pr}_{\bar{M}_k}((\bigsqcap \bar{P}_m)^{i_l i_l}) = \mathrm{pr}_{\bar{M}_k}((\bigsqcap P_m)^{ii}) \doteq \mathrm{pr}_{M_k}((\bigsqcap P_m)^{ii}) = P''^k_m$$

holds for all $m \leq k$. This proves $\bar{\mathcal{L}}_k \doteq \mathcal{L}_k$. The value of $\bar{P}_k$ depends only on $\bar{M}_k$, $\bar{\mathcal{L}}_k$ and $\bar{\mathcal{S}}_k$. It therefore holds that $\bar{P}_k \doteq P_k$.

We have thus shown that Algorithm 11 and Algorithm 9 produce the same output. Since Algorithm 11 is correct, Algorithm 9 must be correct, too. □

In this section we have presented our Model Exploration formalism Algorithm 11 which is based on Algorithm 9. The only motivation for using Algorithm 9 and not Algorithm 10 was to avoid making the algorithm listings even more cluttered than they already are. One might as well apply the same modifications to Algorithm 10 and obtain an equally functional exploration formalism.

Model Exploration can be used for ontology completion in a setting where both an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ and a partial model i_0 are available. In such a setting one would not use a human expert, but a combination of a human expert and a DL reasoner. When a question is asked, one would first use the reasoner to check whether the GCI follows from the existing ontology. Only if this is not the case, will it be presented to the human expert. Upon termination, those GCIs from $\mathcal{B}_6$ that do not already follow from $\mathcal{O}$ are added to $\mathcal{T}$.

Example 6.2. We illustrate Algorithm 11 using the model i from Example 5.3 as the background model. Assume that the initial working model i_0 contains only the first family consisting of Kirk, Luann and Milhouse. Thus we have

$$\Delta_{i_0} = \{\mathsf{Kirk}, \mathsf{Luann}, \mathsf{Milhouse}\}$$

Table 6.4: $\mathbb{K}_0$

	$\bot$	Mother	Father	Female	Male
Kirk			×		×
Luann		×		×	
Milhouse					×

Table 6.5: $\mathbb{K}_1$

	$\bot$	Mother	Father	Female	Male	$\exists c.\top$
Kirk			×		×	×
Luann		×		×		×
Milhouse					×	

and

$$\begin{aligned}
\mathsf{Mother}^{i_0} &= \mathsf{Female}^{i_0} = \{\mathsf{Luann}\} \\
\mathsf{Father}^{i_0} &= \{\mathsf{Kirk}\} \\
\mathsf{Male}^{i_0} &= \{\mathsf{Kirk}, \mathsf{Milhouse}\} \\
\mathsf{c}^{i_0} &= \{(\mathsf{Kirk}, \mathsf{Milhouse}), (\mathsf{Luann}, \mathsf{Milhouse})\}
\end{aligned}$$

1st iteration: The algorithm starts with $P_0 = \emptyset$ and the context $\mathbb{K}_0$ from Table 6.4. We have $\bigsqcap P_0 = \top$ and $\top^{i_0 i_0} = \top$. Therefore the expert is asked if the GCI $\top \sqsubseteq \top$ holds. Obviously, the answer is "yes".[1]

[1] To avoid asking the expert trivial questions one could use a DL-reasoner to check whether P_k is equivalent to $P_k^{i_l i_l}$. We have not added this to Algorithm 11 to keep the pseudo-code more concise. However, checking for equivalence would be necessary in a practical implementation.

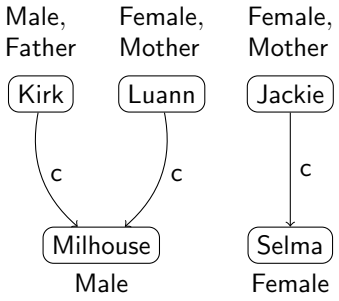

Figure 6.1: Connected Submodel

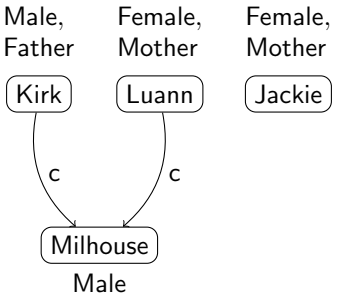

Figure 6.2: Not a Connected Submodel

We compute M_1 by adding the attribute $\exists \mathsf{c}.(\bigsqcap P_0)^{i_0 i_0} = \exists \mathsf{c}.\top$ to M_0. Table 6.5 shows the resulting context $\mathbb{K}_1$.

2nd iteration: We obtain $\mathcal{L}_1 = \emptyset$ and $\mathcal{S}_1 = \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_C\}$. The lectically next set that respects all implications from $\mathcal{L}_1 \cup \mathcal{S}_1$ is $P_1 = \{\mathsf{Mother}\}$. Luann is the only mother in the current working model and Luann's child is a son. We obtain $P_1^{i_0 i_0} = \mathsf{Female} \sqcap \mathsf{Mother} \sqcap \exists \mathsf{c}.\mathsf{Male}$. The next GCI that is presented to the expert is therefore

$$\mathsf{Mother} \sqsubseteq \mathsf{Female} \sqcap \mathsf{Mother} \sqcap \exists \mathsf{c}.\mathsf{Male}.$$

Thus the expert is asked whether all mother are female and have a child that is male. This GCI does not hold in the background model i since Jackie is a mother that has only a daughter. Therefore, the expert rejects the GCI and adds Jackie as a counterexample. She must also add Jackie's daughter Selma because the new model i_1 must be a connected submodel of the background model i (Figure 6.1). Algorithm 11 computes a new GCI based on the new working model i_1. The new GCI

$$\mathsf{Mother} \sqsubseteq \mathsf{Female} \sqcap \mathsf{Mother} \sqcap \exists \mathsf{c}.\top \tag{6.6}$$

is presented to the expert who accepts it. Consequently, the new attribute $\exists \mathsf{c}.(\mathsf{Female} \sqcap \mathsf{Mother} \sqcap \exists \mathsf{c}.\top) = \exists \mathsf{c}.\mathsf{MoC}$ is added.

Without the requirement that working models have to be *connected* submodels of the background model the expert could have added only

Jackie without adding Selma (Figure 6.2). In the model from Figure 6.2 Jackie is a counterexample to the GCI $\mathsf{Mother} \sqsubseteq \exists \mathsf{c}.\top$, i. e. it is not true that all mothers in Figure 6.2 have children. Hence, Algorithm 11 could not have found the GCI (6.6).

We do not look at the *next iterations* in as much details. The following non-trivial GCIs are found:

- $\mathsf{Father} \sqsubseteq \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\mathsf{Male}$ (rejected, Clancy added as counterexample),
- $\mathsf{Father} \sqsubseteq \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\top$ (accepted),
- $\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot$ (accepted),
- $\mathsf{Female} \sqcap \exists \mathsf{c}.\top \sqsubseteq \mathsf{Mother} \sqcap \mathsf{Female} \sqcap \exists \mathsf{c}.\top$ (accepted),
- $\mathsf{Male} \sqcap \exists \mathsf{r}.\top \sqsubseteq \mathsf{Father} \sqcap \mathsf{Male} \sqcap \exists \mathsf{c}.\top$ (accepted),
- $\exists \mathsf{c}.\mathsf{Female} \sqcap \exists \mathsf{c}.\mathsf{Male} \sqsubseteq \bot$ (accepted), and
- $\exists \mathsf{c}.\exists \mathsf{c}.\top \sqsubseteq \bot$ (accepted).

The accepted GCIs are the same GCIs as in Example 5.4. Hence, in this particular example the property of minimal cardinality still holds.

7 ABox Exploration

In order to add a counterexample in Model Exploration the expert needs to provide a connected submodel of the background model. This is not problematic if the background model is fragmented like in Example 5.3. It stands to reason that in practice the more role names occur in a domain the less fragmented will a model representing the domain become. Let us look at the model from Figure 7.1. This model does not have any connected submodels except the model itself. In order to provide a counterexample to even a simple GCI such as $\mathsf{Male} \sqsubseteq \bot$ the expert has to add the entire background model, defeating the purpose of Model Exploration. In this section we introduce ABox Exploration, which allows counterexamples to be provided in an ABox instead of a model. In contrast to the closed-world semantics of a model the open-world semantics of the ABox allow the expert to add just enough information as is needed to describe a counterexample, without unwanted consequences. Except for the way counterexamples are described the setting remains the same as for Model Exploration.

As the logic for the ABox that contains the counterexamples we choose $\mathcal{EL}^{\bot}$ since we would like to remain within the range of $\mathcal{EL}^{++}$. In the first section of this chapter we show that $\mathcal{EL}^{\bot}$ is expressive enough to describe

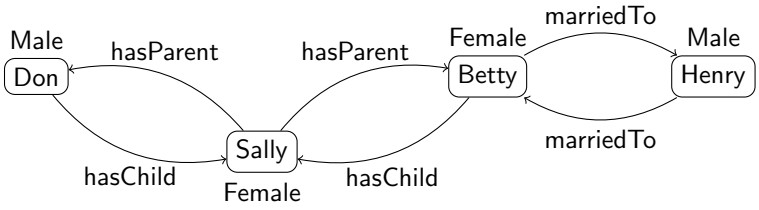

Figure 7.1: A Strongly Connected Background Model

that an individual is a counterexample to a given GCI. In a next step we introduce minimal possible consequences, which can replace model-based most specific concepts in our open-world setting. A large technical part of this chapter answers the question of existence of minimal possible consequences. We later introduce an approximation of minimal possible consequences that has better computational properties. The final section presents the actual exploration algorithms. Some of these results can also be found in [Dis10a].

7.1 Counterexamples in an $\mathcal{EL}^{\bot}$-Ontology

We point out several problems that arise when counterexamples are provided in an ABox instead of a model. First, it is not immediately clear when it makes sense to say that an ABox contains a counterexample. In the first part of this section we present an intuitive type of counterexamples that we call *explicit counterexamples*. We address the issue whether in an open-world setting the expressivity of $\mathcal{EL}^{\bot}$ is sufficient to express that an individual is an explicit counterexample for a given GCI $C \sqsubseteq D$. The second part asks whether it is possible to find an ABox that contains counterexamples for *all* GCIs that do not hold in a given background model i. The last part illustrates the fact that sometimes a GCI $C \sqsubseteq D$ cannot hold in any model of an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, even if no explicit counterexamples are present.

7.1.1 Explicit Counterexamples and Extended Signatures

Intuitively, one would call an ABox individual a a counterexample to the GCI $C \sqsubseteq D$ if a is an instance of C but cannot be an instance of D. We formally introduce the term *explicit counterexample* for such an individual.

Definition 7.1 (Explicit Counterexample)**.** Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology, let $a \in \mathcal{N}_I$ be an individual and $C \sqsubseteq D$ an $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCI. We say that a is an *explicit counterexample* to the GCI $C \sqsubseteq D$ with respect to $\mathcal{O}$ if $\mathcal{O} \models C(a)$ (i. e. all models i of $\mathcal{O}$ satisfy $a^i \in C^i$) and $\mathcal{O} \models \neg D(a)$ (i. e. all models i of $\mathcal{O}$ satisfy $a^i \notin D^i$).

We use an example to illustrate how explicit counterexamples can be provided using only the expressivity of $\mathcal{EL}^{\bot}$.

Example 7.1. In the background model from Figure 7.1 the individual Don is a counterexample to the GCI $\mathsf{Male} \sqsubseteq \exists \mathsf{marriedTo}.\top$. We want to add Don as an individual to an ABox $\mathcal{A}$ and want to make sure that the individual Don is a counterexample to $\mathsf{Male} \sqsubseteq \exists \mathsf{marriedTo}.\top$. It is easy to express that Don is an instance of Male: We can simply add the statement $\mathsf{Male}(\mathsf{Don})$ to $\mathcal{A}$.

Since $\mathcal{EL}^{\bot}$ does not allow for negation it is not trivial to express that Don is not an instance of $\exists \mathsf{marriedTo}.\top$. While this cannot be done using the ABox alone, it can be done using a disjointness statement in the TBox. To this purpose, we extend the signature by adding a new concept name $\mathsf{T}_{\mathsf{Don}}$. We add to the ABox the statement

$$\mathsf{T}_{\mathsf{Don}}(\mathsf{Don})$$

and make sure that $\mathsf{T}_{\mathsf{Don}}$ and $\exists \mathsf{marriedTo}.\top$ are disjoint by adding

$$\mathsf{T}_{\mathsf{Don}} \sqcap \exists \mathsf{marriedTo}.\top \sqsubseteq \bot$$

to $\mathcal{T}$. Because Don is an instance of $\mathsf{T}_{\mathsf{Don}}$ and because we have ensured that $\mathsf{T}_{\mathsf{Don}}$ and $\exists \mathsf{marriedTo}.\top$ are disjoint, we have expressed that Don cannot be an instance of $\exists \mathsf{marriedTo}.\top$.

Clearly, this method to describe counterexamples can be used, not only for the specific GCI, $\mathsf{Male} \sqsubseteq \exists \mathsf{marriedTo}.\top$ but for any GCI $C \sqsubseteq D$. We simply add a new individual a, a new concept name T_a as well as statements $C(a)$, $T_a(a)$ and $D \sqcap T_a \sqsubseteq \bot$. This requires that we are allowed to extend the signature of the current knowledge base $\mathcal{O} = (\mathcal{T}, \mathcal{A})$: We extend the set of individual names $\mathcal{N}_I$ by adding the name a and the set of concept names $\mathcal{N}_C$ by adding the name T_a.

In an exploration setting a background model i is present from the beginning of the exploration, and i is a model of the initial knowledge base $\mathcal{O}_0 = (\mathcal{T}_0, \mathcal{A}_0)$. $\mathcal{O}_0$ uses the set of concept names $\mathcal{N}_C$ and a set of individual names $\mathcal{N}_I$. In such a setting it is problematic to extend the signature of the $\mathcal{O}_0$ by using an extended set of individual names $\mathcal{N}_I \cup \mathcal{N}_I^{\text{new}}$ and a set of concept names $\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}$. Since i is a model of $\mathcal{O}_0$ it maps every individual name $a \in \mathcal{N}_I$ to an element $a^i \in \Delta_i$ and every concept name $A \in \mathcal{N}_C$ to a set $A^i \subseteq \Delta_i$. However, it does not

map individual names from $\mathcal{N}_I^{\text{new}}$ to elements of Δ_i, nor does it map concept names from $\mathcal{N}_C^{\text{new}}$ to subsets of Δ_i. Therefore the background model i cannot be a model of the knowledge base with the extended signature. We need to introduce the notion of a *representation* of an interpretation.

Definition 7.2 (Representation)**.** Let $\mathcal{N}_I$ and $\mathcal{N}_I^{\text{new}}$ be disjoint sets of individual names, let $\mathcal{N}_C$ and $\mathcal{N}_C^{\text{new}}$ be disjoint sets of concept names and let $\mathcal{N}_R$ be a set of role names. Let $i = (\Delta_i, \cdot^i)$ be an $\mathcal{EL}^\bot$-interpretation for the signature $(\mathcal{N}_C, \mathcal{N}_R)$. An ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$ is called a *representation of* i if for every new individual name $a \in \mathcal{N}_I^{\text{new}}$ we can find an element $x_a \in \Delta_i$ and for every new concept name $A \in \mathcal{N}_C^{\text{new}}$ we can find a set $S_A \subseteq \Delta_i$ such that when i is extended to map every new individual name $a \in \mathcal{N}_I^{\text{new}}$ to $a^i = x_a$ and every new concept name $A \in \mathcal{N}_C^{\text{new}}$ to $A^i = S_A$ then i is a model of $\mathcal{O}$.

7.1.2 Completely Describing the Background Model

We have seen that it is possible to describe a counterexample for a given GCI in an $\mathcal{EL}^\bot$-ontology, provided that we allow new concept names to be used. In this section we prove a stronger result, namely that for any given background model i we can find a representation of i that contains an explicit counterexample to *every* GCI that does not hold in i.

Let $i = (\Delta_i, \cdot^i)$ be the background model that uses the signature $(\mathcal{N}_C, \mathcal{N}_R)$. We define an ontology $\mathcal{O}_i = (\mathcal{T}_i, \mathcal{A}_i)$ as follows. For every individual $x \in \Delta_i$ new concept names T_x and F_x are added to the set of concept names, i. e. $\mathcal{N}_C^{\text{new}} = \{T_x, F_x \mid x \in \Delta_i\}$. Furthermore, for every $x \in \Delta_i$ an individual name a_x is added to the set of individual names, i. e. $\mathcal{N}_I^{\text{new}} = \{a_x \mid x \in \Delta_i\}$. Intuitively, we want T_x to represent all properties that x does have and we want F_x to represent all properties that x does not have. The TBox $\mathcal{T}_i$ is defined as

$$\begin{aligned}\mathcal{T}_i = &\{T_x \sqcap F_x \sqsubseteq \bot \mid x \in \Delta_i\} \\ &\cup \{A \sqsubseteq F_x \mid x \in \Delta_i, A \in \mathcal{N}_C, x \notin A^i\} \\ &\cup \Big\{\exists r.\textstyle\bigsqcap S \sqsubseteq F_x \mid r \in \mathcal{N}_R, x \in \Delta_i, S = \{F_y \mid y \in \text{succ}_i^r(x)\}\Big\}.\end{aligned} \tag{7.1}$$

$\mathcal{A}_i$ is defined as

$$\begin{aligned}\mathcal{A}_i = &\{A(a_x) \mid x \in \Delta_i, A \in \text{names}_i(x)\} \\ &\cup \{r(a_x, a_y) \mid (x, y) \in r^i\} \\ &\cup \{T_x(a_x) \mid x \in \Delta_i\}.\end{aligned} \tag{7.2}$$

Lemma 7.1. *$\mathcal{O}_i = (\mathcal{T}_i, \mathcal{A}_i)$ is a representation of i.*

Proof. We define for all $x \in \Delta_i$

- $a_x^i = x$,
- $T_x^i = \{x\}$, and
- $F_x^i = \Delta_i \setminus \{x\}$.

We show that this extension of i is a model of $\mathcal{O}_i = (\mathcal{T}_i, \mathcal{A}_i)$. To this purpose, we need to verify that all statements from $\mathcal{T}_i$ and $\mathcal{A}_i$ hold in i. Here, we only prove this for the most complex type of statement, namely statements of the form $\exists r.\bigsqcap S \sqsubseteq F_x$, where $r \in \mathcal{N}_R$, $x \in \Delta_i$ and $S = \{F_y \mid y \in \text{succ}_i^r(x)\}$. Let $z \in \Delta_i$ be an individual that satisfies $z \in (\exists r.\bigsqcap S)^i$. This implies that there is some r-successor z' of z such that $z' \in F_y^i$ holds for all $y \in \text{succ}_i^r(x)$. By definition it holds that $F_y^i = \Delta_i \setminus \{y\}$ and therefore we obtain $z' \neq y$ for all $y \in \Delta_i$ that are r-successors of x. Hence z' cannot be an r-successor of x and thus $z \neq x$ holds. From $F_x^i = \Delta_i \setminus \{x\}$ it follows that $z \in F_x^i$. Therefore $\exists r.\bigsqcap S \sqsubseteq F_x$ holds in i.

One can prove in a similar fashion that all other statements from $\mathcal{O}_i$ hold in i. We obtain that the extended version of i is a model of $\mathcal{O}_i$ and thus $\mathcal{O}_i$ is a representation of i. □

We have shown that $\mathcal{O}_i$ is a representation of i. We now prove that it completely describes i, i. e. that it contains an explicit counterexample to every GCI that does not hold in i. The following technical lemma helps us with that.

Lemma 7.2. *Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and let $x \in \Delta_i$ be an individual. Let j be a model of $\mathcal{O}_i$. Then $x \in C^i$ holds iff $a_x^j \in C^j$ holds.*

Proof. The case $C = \bot$ is trivial. We assume that $C = (A_C, \mathcal{T}_C)$ is an $\mathcal{EL}_{\text{gfp}}$-concept description. We first prove that $x \in C^i$ implies $a_x^j \in C^j$ by constructing a simulation. From $x \in C^i$ and Lemma 2.2 it follows that there is a simulation Z from C to x in i. We prove that the relation Z_1 defined as

$$Z_1 = \{(B, a_y^j) \mid (B, y) \in Z\}$$

is a simulation from C to a_x^j in j. *(S1′)* Let $(B, a_y^j) \in Z_1$ be a pair in Z_1. Then $(B, y) \in Z$ holds. Since Z is a simulation from C to x in i it holds that $\text{names}_{\mathcal{T}_C}(B) \subseteq \text{names}_i(y)$. Let $A \in \text{names}_i(y)$ be a concept name. Then $\mathcal{A}_i$ contains the statement $A(a_y)$. Since j is a model of $\mathcal{O}_i$ it holds that $a_y^j \in A^j$ and thus $A \in \text{names}_j(a_y^j)$. This proves $\text{names}_i(y) \subseteq \text{names}_j(a_y)$. Together with $\text{names}_{\mathcal{T}_C}(B) \subseteq \text{names}_i(y)$ we obtain $\text{names}_{\mathcal{T}_C}(B) \subseteq \text{names}_j(a_y^j)$. This proves (S1′) for Z_1. *(S2′)* Let $(B, a_y^j) \in Z_1$ be a pair in the relation Z_1 and let $B' \in \text{succ}_{\mathcal{T}_C}^r(B)$ be an r-successor of B. According to the definition of Z_1 it holds that $(B, y) \in Z$. Since Z is a simulation satisfying (S2′) there must be some $y' \in \text{succ}_i^r(y)$ that satisfies $(B', y') \in Z$. $(B', a_{y'}^j) \in Z_1$ follows immediately from the definition of Z_1. It remains to show that $a_{y'}^j \in \text{succ}_j^r(a_y^j)$ holds. Because of $y' \in \text{succ}_i^r(y)$ the ABox $\mathcal{A}_i$ contains a statement $r(a_y, a_{y'})$. Since j is a model of $\mathcal{O}_j$ this implies $a_{y'}^j \in \text{succ}_i^r(a_y^j)$. This proves (S2′) for Z_1. *(S3′)* Since Z contains the pair (A_C, x) we readily obtain $(A_C, a_x^j) \in Z_1$.

We have thus shown that Z_1 is a simulation from C to a_x^j in j. Lemma 2.2 yields $a_x^j \in C^j$. We now prove the reverse direction where we assume that $a_x^j \in C^j$ holds and show that this implies $x^i \in C^i$. We first prove that

$$Z_2 = \{(y, z) \mid y \notin F_z^j\}$$

is a simulation from j to i. *(S1)* Let $(y, z) \in Z_2$ be a pair from Z_2 and let A be a concept name that satisfies $y \in A^j$. Assume that $z^i \notin A^i$ holds. Then $\mathcal{T}_i$ contains the statement $A \sqsubseteq F_z$. Since j is a model of $\mathcal{O}_i$ and since $y \in A^j$ holds we obtain $y \in F_z^j$. This contradicts $(y, z) \in Z_2$ and therefore the assumption $z^i \notin A^i$ can be refuted. We have thus shown that $y \in A^j$ implies $z \in A^i$ and therefore $\text{names}_j(y) \subseteq \text{names}_i(z)$.

(S2) Let $(y, z) \in Z_2$ be a pair from the Z_2, let $r \in \mathcal{N}_R$ be a role name and let $y' \in \text{succ}_j^r(y)$ be an r-successor of y. Assume that $(y', z') \notin Z_2$ holds for all $z' \in \text{succ}_i^r(z)$. Then the definition of Z_2 yields $y' \in F_{z'}^j$ for all $z' \in \text{succ}_i^r(z)$. We obtain $y' \in (\bigsqcap S)^j$, where $S = \{F_{z'} \mid z' \in \text{succ}_i^r(z)\}$.

Therefore $y \in (\exists r.\bigsqcap S)^j$ holds. On the other hand, $\mathcal{T}_i$ contains the statement $\exists r.\bigsqcap S \sqsubseteq F_z$. Since j is a model of $\mathcal{O}_i$ this yields $y \in F_z^j$. This contradicts $(y,z) \in Z_2$. Therefore the assumption that $(y',z') \notin Z_2$ holds for all $z' \in \mathrm{succ}_i^r(z)$ must be false. This proves (S2) for Z_2 We have thus shown that Z_2 is a simulation from j to i. It is readily verified that Z_2 contains the pair (a_x^j, x).

Lemma 2.2 and $a_x^j \in C^j$ yield that there is a simulation Z_3 from C to a_x^j in j. Since the concatenation of two simulations is also a simulation the relation $Z_2 \circ Z_3$ must be a simulation from C to x in i. We obtain $x^i \in C^i$ from Lemma 2.2. □

Theorem 7.3. *If $C \sqsubseteq D$ is a GCI that does not hold in i then $\mathcal{O}_i$ contains an explicit counterexample for $C \sqsubseteq D$.*

Proof. Since $C \sqsubseteq D$ does not hold in i there exists some $x \in C^i$ such that $x \notin D^i$. We show that the corresponding ABox-individual a_x is an explicit counterexample to $C \sqsubseteq D$. Let j_1 be a model of $\mathcal{O}_i$. From $x \in C^i$ and Lemma 7.2 it follows that $a_x^{j_1} \in C^{j_1}$ holds. $\mathcal{O}_i \models C(a_x)$ follows because j_1 is an arbitrary model of $\mathcal{O}_i$.

Conversely, assume that there is a model j_2 of $\mathcal{O}_i$ which satisfies $a_x^{j_2} \in D^{j_2}$. Then Lemma 7.2 implies $x \in D^i$. This contradicts $x \notin D^i$ and therefore such a model j_2 cannot exist. Hence, $\mathcal{O}_i \models \neg D(a_x)$ holds. Together with $\mathcal{O}_i \models C(a_x)$ we obtain that a_x is an explicit counterexample to $C \sqsubseteq D$. □

7.1.3 Counterexamples Need not be Explicit

A perhaps surprising fact is that in an $\mathcal{EL}^{\bot}$-ontology counterexamples need not be explicit. If an ontology $\mathcal{O}$ does not contain an explicit counterexample to a GCI $C \sqsubseteq D$ then it is still possible that $C \sqsubseteq D$ does not hold in any model of $\mathcal{O}$. The following example illustrates this.

Example 7.2 (Counterexamples Need not be Explicit)**.** Let us look at the following ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ containing two individuals, a mother Peggy and an Infant. The ontology contains the following knowledge.

$$\mathcal{T} = \{\mathsf{T}_{\mathsf{Peggy}} \sqcap \exists \mathsf{hasChild}.\exists \mathsf{hasChild}.\top \sqsubseteq \bot,$$
$$\mathsf{T}_{\mathsf{Infant}} \sqcap \exists \mathsf{hasChild}.\top \sqsubseteq \bot\}$$

and

$$\mathcal{A} = \{\mathsf{T}_{\mathsf{Peggy}}(\mathsf{Peggy}), \mathsf{Female}(\mathsf{Peggy}), \\ \mathsf{T}_{\mathsf{Infant}}(\mathsf{Infant}), \mathsf{hasChild}(\mathsf{Peggy}, \mathsf{Infant})\}$$

The TBox axioms ensure that Peggy does not have any grandchildren, while the Infant does not have any children. The ABox yields that Peggy is Female and the Infant's mother. The ontology does not contain the information whether the Infant is Female or not.

We consider the GCI $\mathsf{Female} \sqsubseteq \exists \mathsf{hasChild}.\mathsf{Female}$. Infant is not an explicit counterexample since it is not an instance of Female (cf. Figures 7.2 and 7.4). Peggy is not an explicit counterexample, either. Even though Peggy is an instance of Female, there are models i in which $\mathsf{Peggy} \in (\exists \mathsf{hasChild}.\mathsf{Female})^i$ does hold (cf. Figures 7.3 and 7.4). Thus there are no explicit counterexamples to $\mathsf{Female} \sqsubseteq \exists \mathsf{hasChild}.\mathsf{Female}$ in $\mathcal{O}$.

Suprisingly, every model of $\mathcal{O}$ must still contain a counterexample. Assume that i_1 is a model of $\mathcal{O}$ in which Peggy^{i_1} does not have any hasChild-successors that are in Female^{i_1}. Then $\mathsf{Peggy}^{i_1} \in \mathsf{Female}^{i_1}$ holds, but also $\mathsf{Peggy}^{i_1} \notin \exists \mathsf{hasChild}.\mathsf{Female}$ holds. Thus Peggy^{i_1} serves as a counterexample in the model i_1.

Conversely, assume that i_2 is a model of $\mathcal{O}$ in which Peggy^{i_2} does have a hasChild-successor $x \in \mathsf{Female}^{i_2}$. Peggy^{i_2} does not have any grandchildren because of the first statement in $\mathcal{T}$. Thus x cannot have any hasChild-successors in i_2 and in particular $x \notin (\exists \mathsf{hasChild}.\mathsf{Female})^{i_2}$ holds. Thus x serves as a counterexample in i_2.

We have thus shown that while there is no explicit counterexample present in $\mathcal{O}$ the GCI $\mathsf{Female} \sqsubseteq \exists \mathsf{hasChild}.\mathsf{Female}$ still cannot hold in any model of $\mathcal{O}$. Every model i of $\mathcal{O}$ must contain a counterexample, but that counterexample can be Peggy^i (cf. Figure 7.2) or Infant^i (cf. Figure 7.3) or even an unnamed individual $x \in \Delta_i$ as in Figure 7.4. Hence, there is no model of $\mathcal{O}$ in which the GCI $\mathsf{Female} \sqsubseteq \exists \mathsf{hasChild}.\mathsf{Female}$ holds.

Definition 7.3. Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology, and $C \sqsubseteq D$ an $\mathcal{EL}^{\bot}_{\mathrm{gfp}}$-GCI. We say that $C \sqsubseteq D$ is *refuted* by $\mathcal{O}$ if there is no model i of $\mathcal{O}$ in which $C \sqsubseteq D$ holds.

Clearly, if $\mathcal{O}$ contains an explicit counterexample to the GCI $C \sqsubseteq D$ then $C \sqsubseteq D$ is refuted by $\mathcal{O}$. Example 7.2 shows that the converse is

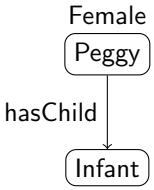

Figure 7.2: Peggy Serves as Counter-example

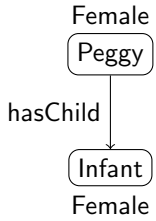

Figure 7.3: Infant Serves as Counter-example

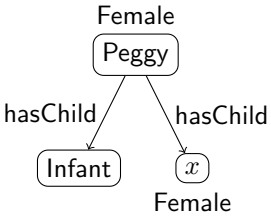

Figure 7.4: x Serves as Counterexample

not true. We can use a reasoner to check whether $C \sqsubseteq D$ is refuted by $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ using consistency checking. We could simply have the reasoner perform a consistency check on the ontology $\mathcal{O}' = (\mathcal{T}', \mathcal{A})$ where $\mathcal{T}' = \mathcal{T} \cup \{C \sqsubseteq D\}$. If $\mathcal{O}'$ is consistent, then $C \sqsubseteq D$ is not refuted by $\mathcal{O}$.

7.2 Minimal Possible Consequences and Their Approximations

7.2.1 Definitions

When developing knowledge base completion formalisms we strive for completeness with respect to the GCIs that hold in a background model i. In this section we consider a setting where we are given a representation $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ of i. We can distinguish three types of GCIs. First, there are the GCIs that follow from $\mathcal{O}$. These GCIs hold in all models of $\mathcal{O}$ and – provided they do not use any of the new concept names – also in the background model i. It is not interesting to present such GCIs to an expert since the knowledge about them is already present in $\mathcal{O}$ and can be obtained using subsumption reasoning. At the other end of the spectrum are the GCIs that do not hold in any model of $\mathcal{O}$. These are the GCIs that are refuted by $\mathcal{O}$ in the sense of Definition 7.3. They are also not interesting, since the information that they cannot hold is

already implicitly present in $\mathcal{O}$.

The GCIs that are relevant for knowledge base completion are those that fall into neither of the two categories, namely those that neither follow from $\mathcal{O}$ nor are refuted by $\mathcal{O}$.

Definition 7.4 (Possible Consequence)**.** Let C and D be $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions and $\mathcal{O}$ an $\mathcal{EL}^{\bot}$-ontology. If $C \sqsubseteq D$ is not refuted by $\mathcal{O}$ then D is called a *possible consequence of* C with respect to $\mathcal{O}$. Conversely, we say that D is a *certain consequence* of C if $C \sqsubseteq D$ follows from $\mathcal{O}$.

We denote the set of all possible consequences of C with respect to $\mathcal{O}$ by $\mathfrak{pc}_{\mathcal{O}}(C)$. If D is a possible consequence of C and i is a model of $\mathcal{O}$ such that $C \sqsubseteq D$ holds in i then we call i a *witness model* for $C \sqsubseteq D$.

We have seen in Section 7.1.1 that it is sometimes necessary to extend the set of concept names in order to define counterexamples. This motivates why we introduce possible consequences also for a restricted signature: Let $\mathcal{O}$ be an ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$. We denote the set of all $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions C over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ that are possible consequences with respect to $\mathcal{O}$ by $\mathfrak{pc}^{\mathcal{N}_C}_{\mathcal{O}}(C)$.

When we present a GCI $C \sqsubseteq D$ as a question to the expert the right-hand side D should be a possible consequence of the left-hand side C, but not a certain consequence. However, it is not a good strategy to use any possible consequence of C. Like in the previous chapters, we want to make sure that, once the expert has accepted $C \sqsubseteq D$, no other GCIs with C as their left-hand side need to be asked. Therefore, we want to ensure that the right-hand side is as specific as possible.

Definition 7.5 (Minimal Possible Consequence)**.** Let $\mathcal{O}$ be an $\mathcal{EL}^{\bot}$-ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$. Let C be a $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description over the restricted signature $(\mathcal{N}_C, \mathcal{N}_R)$.

The minimal elements of $\mathfrak{pc}^{\mathcal{N}_C}_{\mathcal{O}}(C)$ with respect to $\sqsubseteq$ are called *minimal possible consequences* of C with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$. We denote the set of minimal possible consequences of C with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$ by $\mathfrak{mpc}^{\mathcal{N}_C}_{\mathcal{O}}(C)$. If no set of concept names is specified then we allow the full set of concept names $\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}$ used in $\mathcal{O}$.

Notice that we use $\mathcal{EL}^{\bot}$ for the ontology and $\mathcal{EL}^{\bot}_{\text{gfp}}$ for the minimal possible consequences. Like in Model Exploration, the idea is that

cyclic concept descriptions are used temporarily during the exploration, but should be removed upon termination using the construction from Section 5.3. Cyclic concept descriptions should neither be nor become part of the actual ontology, since they are currently not supported by reasoners.

In an open world setting minimal possible consequences are the equivalent notion to model-based most specific concepts in a closed world setting.[1] Unfortunately, unlike model-based most specific concepts, minimal possible consequences are not unique up to equivalence. We demonstrate this in the following example.

Example 7.3 (Minimal Possible Consequences are not Unique)**.** Minimal possible consequences are not unique, even in a very simple setting where the set of role names $\mathcal{N}_R$ is empty. Let $\mathcal{N}_C = \{A, B, C\}$ be the set of concept names. We consider the TBox

$$\mathcal{T} = \{A \sqcap B \sqcap C \sqsubseteq \bot\}$$

and the ABox

$$\mathcal{A} = \{A(a)\}.$$

We look for a minimal possible consequence of A with respect to $(\mathcal{T}, \mathcal{A})$ allowing the full set of concept names. Clearly, since every model i has to contain an individual $a^i \in A^i$ the bottom concept $\bot$ is not a possible consequence of A. Also $A \sqsubseteq A \sqcap B \sqcap C$ cannot hold in any model i of $\mathcal{O}$ because of the TBox statement $A \sqcap B \sqcap C \sqsubseteq \bot$. However, both $A \sqsubseteq A \sqcap B$ and $A \sqsubseteq A \sqcap C$ are not refuted by $\mathcal{O}$ and thus both $A \sqcap B$ and $A \sqcap C$ are minimal possible consequences of A with respect to $\mathcal{O}$.

Example 7.4 (Minimal Possible Consequences can be Large)**.** Let n be a natural number. Let the set of concept names be $\mathcal{N}_C = \{A, B_{10}, B_{11}, B_{20}, B_{21}, \ldots, B_{n0}, B_{n1}\}$ and let the set of role names be $\mathcal{N}_R = \{r\}$. We

[1] Note that, despite the similarity in names, most specific concepts for ABoxes have a different flavour. They express the properties that an ABox-individual *must* have, and are thus more closely related to certain consequences than to possible consequences.

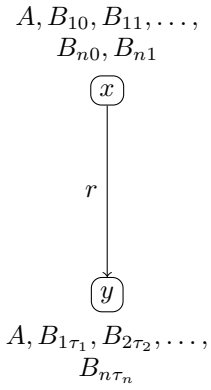

Figure 7.5: Witness Model for C_τ

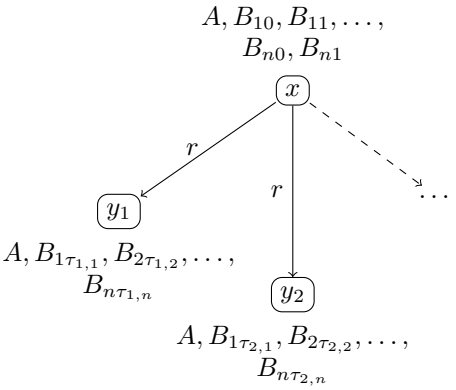

Figure 7.6: Witness Model for D

consider the TBox

$$\begin{aligned}\mathcal{T} = \{&A \sqcup \exists r.A \sqsubseteq \bot,\\ &A \sqcap \exists r.\exists r.\top \sqsubseteq \bot,\\ &A \sqcap \exists r.(B_{10} \sqcap B_{11}) \sqsubseteq \bot,\\ &A \sqcap \exists r.(B_{20} \sqcap B_{21}) \sqsubseteq \bot,\\ &\vdots\\ &A \sqcap \exists r.(B_{n0} \sqcap B_{n1}) \sqsubseteq \bot\}\end{aligned}$$

and the ABox

$$\mathcal{A} = \{A(a)\}.$$

For every binary vector $\tau = (\tau_1, \ldots, \tau_n) \in \{0,1\}^n$ the concept description $C_\tau = \exists r.\bigsqcap S_\tau$, where $S_\tau = \{B_{k\tau_k} \mid k \in \{1, \ldots, n\}\}$, is a possible consequence of A with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ (cf. Figure 7.5 for a witness model). Also the conjunction

$$D = \bigsqcap \mathcal{N}_C \sqcap \bigsqcap_{\tau \in \{0,1\}^n} C_\tau \tag{7.3}$$

is a possible consequence of A with respect to $\mathcal{O}$ (cf. Figure 7.6 for a witness model). In every model i of $\mathcal{O}$ there is at least one individual in A^i, namely a^i. An r-successor of a^i cannot have an r-successor itself, nor can it be an element of A^i nor an element of $(B_{k0} \sqcap B_{k1})^i$ for some $k \in \{1, \ldots, n\}$. All of this would contradict one of the TBox-statements. Hence D as defined in (7.3) is a minimal possible consequence of A with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. One can even verify that it is the only minimal possible consequence of A. D is exponentially large in n and there is no equivalent $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description that is more succinct. This demonstrates that an exponential blowup in size cannot be avoided when minimal possible consequences are computed.

7.2.2 Existence

In this section we prove that in $\mathcal{EL}^{\bot}$-ontologies minimal possible consequences always exist. The proof also yields a construction, which is, however, not very efficient. Matters are complicated by the facts that minimal possible consequences are not unique and that they can be exponentially large. We divide the proof into three parts, starting with simpler settings and then gradually moving towards the general setting, eventually leading to the main result of this section, Theorem 7.18. Readers who are only interested in the ABox Exploration algorithms may want to skip these proofs.

Restricted Canonical Models

Eventually, we want to prove existence of minimal possible consequences for the general setting where a TBox and an ABox are present and where new concept names have been added to the TBox. For now, we make several simplifications. We are not looking for minimal possible consequences of a complex $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description, but of a concept name $A \in \mathcal{N}_C$. Second, we assume that $\mathcal{T}$ is a general $\mathcal{EL}^{\bot}$-TBox over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ and that $\mathcal{T}$ is in the following normal form: Every axiom in $\mathcal{T}$ is of one of the forms

$$\begin{aligned} E \sqcap F &\sqsubseteq G \\ E &\sqsubseteq \exists r.F \\ \exists r.E &\sqsubseteq G \end{aligned} \tag{7.4}$$

where $r \in \mathcal{N}_R$ is a role-name, and $E, F \in \mathcal{N}_C \cup \{\top\}$ and $G \in \mathcal{N}_C \cup \{\bot\}$ are concept descriptions. Finally, we require that no ABox is present. Our goal is to prove that $\mathtt{mpc}_{\mathcal{O}}^{\mathcal{N}_C}(A)$ is not empty, where $\mathcal{O} = (\mathcal{T}, \emptyset)$ and $\mathcal{N}_C$ is the full set of concept names used in $\mathcal{T}$ in its normal form.

By asking a reasoner whether $\mathcal{T} \cup \{A \sqsubseteq \bot\}$ is consistent we can check whether $\bot$ is a possible consequence of A with respect to $\mathcal{O} = (\mathcal{T}, \emptyset)$. If it is, then a minimal possible consequence obviously exists. In the following we assume that $\bot$ is not a minimal possible consequence. In this case minimal possible consequences, provided that they exist, must be $\mathcal{EL}_{\text{gfp}}$-concept descriptions. The idea of our proof is to provide a construction, not for the concept description itself, but for one of its witness models. Intuitively, a minimal possible consequence E of A is a maximally "complex" $\mathcal{EL}_{\text{gfp}}$-concept description such that $\mathcal{T} \cup \{A \sqsubseteq E\}$ remains consistent. A first intuition is that the witness model should then also be as "complex" as possible. This reasoning motivates that a *canonical model* of $\mathcal{T}$, i. e. a model κ of $\mathcal{T}$ such that there is a simulation from every model i of $\mathcal{T}$ to κ, is a witness model of a minimal possible consequence. Unfortunately, this is not the case in situations where minimal possible consequences are not unique.[2] This is why we introduce *restricted models*.

Definition 7.6 (Restricted Model)**.** Let $\mathcal{O}$ be an ontology and i a model of $\mathcal{O}$. Let $R \subseteq \mathcal{N}_C$ be a set of concept names. We say that i is *restricted* with respect to $A \to R$ if $A \sqsubseteq \bigsqcap R$ holds in i. We say that i is *maximally restricted* with respect to $A \to R$ if i is restricted with respect to $A \to R$, and there is no strict superset $\bar{R} \supsetneq R$ such that i is restricted with respect to $A \to \bar{R}$.

R is called a *restriction set* of A with respect to $\mathcal{T}$ if there is a model i of $\mathcal{T}$ such that i is maximally restricted with respect to $A \to R$.

Let R be a restriction set of A with respect to $\mathcal{T}$. Instead of constructing the canonical model of $\mathcal{T}$ we construct a model that is canonical among all models that are restricted with respect to $A \to R$, i. e. a

[2] An example where the canonical model is not the witness model of minimal possible consequence is Example 7.3. In this example one canonical model would be $i = (\{x, y, z\}, \cdot^i)$, where $A^i = \{x, y\}$, $B^i = \{x, z\}$, $C^i = \{y, z\}$ and $a^i = x$. It is not a witness for either of the minimal possible consequences of A. In this example an ABox is present, but examples without ABoxes are also not difficult to construct.

model κ_R such that κ_R is restricted with respect to $A \to R$ and for every model i that is restricted with respect to $A \to R$ there is a simulation from i to κ_R.

Definition 7.7 (Candidate Namesets)**.** A set of concept names $U \subseteq \mathcal{N}_C$ is called a *candidate nameset* with respect to $\mathcal{T}$ and $A \to R$ if

- $A \notin U$ or $R \subseteq U$,
- for all TBox statements $E \sqcap F \sqsubseteq G \in \mathcal{T}$ it holds that $E \in U \cup \{\top\}$ and $F \in U \cup \{\top\}$ implies $G \in U$.

Let $i = (\Delta_i, \cdot^i)$ be a model of $\mathcal{T}$ that is restricted with respect to $A \to R$. Then for all individuals $x \in \Delta_i$ the set $\text{names}_i(x)$ is a candidate nameset with respect to $\mathcal{T}$ and $A \to R$. The converse is not true, i. e. not all candidate namesets occur in an actual model. To find those candidate namesets that do occur in a model we need to look at the other two types of TBox statements from (7.4). Furthermore, x can have an r-successor if and only if $\text{can}^r_{\mathcal{T}}(U)$ contains $\top$.

The TBox axioms of the form $E \sqsubseteq \exists r.F \in \mathcal{T}$ tell us something about the names that must occur in some r-successor of x. Clearly, if $E \in \text{names}_i(x)$ then there must be some r-successor y of x with $F \in \text{names}_i(y)$. We define for every set $U \subseteq \mathcal{N}_C$

$$\text{must}^r_{\mathcal{T}}(U) = \big\{F \mid E \sqsubseteq \exists r.F \in \mathcal{T}, E \in U \cup \{\top\}\big\}. \tag{7.5}$$

Here $\top \in \text{must}^r_{\mathcal{T}}(U)$ implies that every individual x with $\text{names}_i(x) = U$ must have at least one r-successor. The third type of TBox axioms is of the form $\exists r.E \sqsubseteq G \in \mathcal{T}$. These statements tell us which names an r-successor of x cannot have. If $G \notin \text{names}_i(x)$ holds then $E \notin \text{names}_i(y)$ must hold for all r-successors $y \in \text{succ}^r_i(x)$ or otherwise the statement $\exists r.E \sqsubseteq G \in \mathcal{T}$ would be violated. Hence, the concept names that cannot occur in an r-successor of x are

$$\big\{E \in \mathcal{N}_C \mid \exists r.E \sqsubseteq G \in \mathcal{T}, G \notin \text{names}_i(x)\big\}$$

For a set $U \subseteq \mathcal{N}_C$ we define

$$\text{can}^r_{\mathcal{T}}(U) = (\mathcal{N}_C \cup \{\top\}) \setminus \big\{E \mid \exists r.E \sqsubseteq G \in \mathcal{T}, G \notin U\big\}. \tag{7.6}$$

Then $\text{can}^r_{\mathcal{T}}(U)$ contains those concept names that can occur as the name of an r-successor of an individual $x \in \Delta_i$, if x satisfies $\text{names}_i(x) = U$.

To determine whether a given set $\mathcal{U} \subseteq 2^{\mathcal{N}_C}$ gives rise to a model of $\mathcal{T}$ we introduce the *model-conditions*. We say that $\mathcal{U}$ satisfies the model-conditions with respect to $\mathcal{T}$ and $A \to R$ if

- all sets $U \in \mathcal{U}$ are candidate namesets with respect to $\mathcal{T}$ and $A \to R$, and
- for all $U \in \mathcal{U}$ and all role names $r \in \mathcal{N}_R$ it holds that

$$\text{must}^r_{\mathcal{T}}(U) \subseteq \bigcup \big\{ V \in \mathcal{U} \mid V \cup \{\top\} \subseteq \text{can}^r_{\mathcal{T}}(U) \big\}.$$

If $\mathcal{U} \subseteq 2^{\mathcal{N}_C}$ and $\mathcal{V} \subseteq 2^{\mathcal{N}_C}$ both satisfy the model-conditions then their union $\mathcal{U} \cup \mathcal{V}$ also satisfies the model-conditions. Therefore for any TBox $\mathcal{T}$ and any restriction $A \to R$ there exists a greatest set $\mathcal{U}_{\max} \in 2^{\mathcal{N}_C}$ (with respect to set inclusion) that satisfies the model-conditions, namely the union of all subsets of $2^{\mathcal{N}_C}$ that satisfy the model-conditions.

It is possible to show that the intersection of two candidate namesets U and V is also a candidate nameset. Furthermore, it holds that $\text{must}^r_{\mathcal{T}}(U \cap V) = \text{must}^r_{\mathcal{T}}(U) \cap \text{must}^r_{\mathcal{T}}(V)$ and $\text{can}^r_{\mathcal{T}}(U \cap V) = \text{can}^r_{\mathcal{T}}(U) \cap \text{can}^r_{\mathcal{T}}(V)$. This can be used to show that $\mathcal{U}_{\max}$ is closed under intersection.

Definition 7.8 (Canonical Restricted Model)**.** Let $\mathcal{U}_{\max}$ be the greatest subset of $2^{\mathcal{N}_C}$ that satisfies the model conditions with respect to $\mathcal{T}$ and $A \to R$. Then the *canonical restricted model* $\kappa_R = (\Delta_{\kappa_R}, \cdot^{\kappa_R})$ is defined as

- $\Delta_{\kappa_R} = \mathcal{U}_{\max}$,
- for all $B \in \mathcal{N}_C$ we define $B^{\kappa_R} = \big\{ U \in \mathcal{U}_{\max} \mid B \in U \big\}$, and
- for all $r \in \mathcal{N}_R$ we define $r^{\kappa_R} = \big\{ (U, V) \in \mathcal{U}_{\max} \times \mathcal{U}_{\max} \mid V \cup \{\top\} \subseteq \text{can}^r_{\mathcal{T}}(U) \big\}$.

Since the individuals in Δ_{κ_R} are subsets of $\mathcal{N}_C$ we violate the convention that individuals be denoted by lower case letters. Also notice that $\text{names}_{\kappa_R}(U) = U$ holds for all $U \in \mathcal{U}_{\max}$.

Lemma 7.4. *If R is a restriction set for A with respect to $\mathcal{T}$ then*

1. $\mathcal{U}_{\max}$ *is not empty and in particular* $R \in \mathcal{U}_{\max}$ *holds, and*

2. κ_R *is a model of* $\mathcal{T}$ *that is restricted with respect to* $A \to R$.

Proof. 1. Since R is a restriction set for A with respect to $\mathcal{T}$ there must be a model $i = (\Delta_i, \cdot^i)$ of $\mathcal{T}$ that is maximally restricted with respect to $A \to R$. The set of all namesets that occur in i, i. e. the set $\{\text{names}_i(x) \mid x \in \Delta_i\}$, satisfies the model-conditions. Hence it holds that $\{\text{names}_i(x) \mid x \in \Delta_i\} \subseteq \mathcal{U}_{\max}$. Since Δ_i is not empty this implies that $\mathcal{U}_{\max}$ is not empty either. Furthermore, i is maximally restricted with respect to $A \to R$ and thus $\bigcap_{a \in A^i} \text{names}_i(a) = R$ follows. Since $\mathcal{U}_{\max}$ is closed under intersection we obtain $R \in \mathcal{U}_{\max}$.

2. To show that κ_R is a model of $\mathcal{T}$ we need to verify that all statements from $\mathcal{T}$ hold in κ_R. We look at the three types of statements separately. *Type* $E \sqcap F \sqsubseteq G \in \mathcal{T}$*:* Let $U \in \mathcal{U}_{\max}$ be an individual such that $U \in (E \sqcap F)^{\kappa_R} = E^{\kappa_R} \cap F^{\kappa_R}$. If both E and F are concept names then by Definition 7.8 this is equivalent to $E \in \text{names}_{\kappa_R}(U) = U$ and $F \in \text{names}_{\kappa_R}(U) = U$. Since $\mathcal{U}_{\max}$ contains only candidate namesets U must be a candidate nameset. Then $E \in U$ and $F \in U$ implies $G \in U = \text{names}_{\kappa_R}(U)$ and therefore $U \in G^{\kappa_R}$. The case where $E = \top$ or $F = \top$ can be treated analogously.

Type $\exists r.E \sqsubseteq G \in \mathcal{T}$*:* Let $U \in \mathcal{U}_{\max}$ be an individual that satisfies $U \in (\exists r.E)^{\kappa_R}$. Then there must be some r-successor $V \in \text{succ}^r_{\kappa_R}(U)$ that satisfies $V \in E^{\kappa_R}$. Definition 7.8 and $V \in \text{succ}^r_{\kappa_R}(U)$ imply that $V \cup \{\top\} \subseteq \text{can}^r_{\mathcal{T}}(U)$ holds. Definition 7.8 and $V \in E^{\kappa_R}$ imply that $E \in V \cup \{\top\}$ holds. Therefore we obtain $E \in \text{can}^r_{\mathcal{T}}(U)$. Finally, (7.6) and $\exists r.E \sqsubseteq G \in \mathcal{T}$ yield $G \in U$. Thus $U \in G^{\kappa_R}$ follows from Definition 7.8.

Type $E \sqsubseteq \exists r.F \in \mathcal{T}$*:* Let $U \in \mathcal{U}_{\max}$ be an individual satisfying $U \in E^{\kappa_R}$. Then by Definition 7.8 it holds that $E \in U \cup \{\top\}$. From (7.5) we obtain $F \in \text{must}^r_{\mathcal{T}}(U)$. Since $\mathcal{U}_{\max}$ satisfies the model-conditions there must be some $V \in \mathcal{U}_{\max}$ satisfying $F \in V$ and $V \subseteq \text{can}^r_{\mathcal{T}}(U)$. We obtain from Definition 7.8 that $V \in F^{\kappa_R}$ and $V \in \text{succ}^r_{\kappa_R}(U)$ hold. This proves $U \in (\exists r.F)^{\kappa_R}$.

We have shown that all three types (7.4) of statements from $\mathcal{T}$ hold in κ_R. Therefore κ_R is a model of $\mathcal{T}$. All elements of $\mathcal{U}_{\max}$ are candidate namesets. Thus if some set $U \in \mathcal{U}_{\max}$ satisfies $A \in U = \text{names}_{\kappa_R}(U)$ then $R \subseteq \text{names}_{\kappa_R}(U)$ holds. This proves that κ_R is restricted with respect to $A \to R$. □

We have shown that for any restriction set R the canonical restricted model κ_R is a model of $\mathcal{T}$. The name *canonical* restricted model suggests that there should be a simulation from each restricted model of T to κ_R. The following lemma proves this.

Lemma 7.5. *If R is a restriction set for A with respect to $\mathcal{T}$ and i is a model of $\mathcal{T}$ that is restricted with respect to $A \to R$ then*

$$Z_i = \{(x, \text{names}_i(x) \mid x \in \Delta_i\}$$

is a simulation from i to κ_R.

Proof. (S1) Let $(x, U) \in Z_i$ be a pair in Z_i. Then by definition of Z_i it holds that $U = \text{names}_i(x)$. It holds that $\text{names}_{\kappa_R}(U) = U = \text{names}_i(x)$ which proves (S1) for Z_i.

(S2) Let $(x, U) \in Z_i$ be a pair in Z_i, let $r \in \mathcal{N}_R$ be a role name and $y \in \text{succ}_i^r(x)$ be an r-successor of x in i. We define $V = \text{names}_i(y)$. Then $(y, V) \in Z_i$ holds by definition. We still need to show that $V \in \text{succ}_{\kappa_R}^r(U)$ holds, i. e. we need to prove $V \cup \{\top\} \subseteq \text{can}_{\mathcal{T}}^r(U)$. We assume the contrary. Let $E \in V \cup \{\top\}$ be a concept description that satisfies $E \notin \text{can}_{\mathcal{T}}^r(U)$. We obtain from (7.6) that there is some GCI $\exists r.E \sqsubseteq G \in \mathcal{T}$ such that $G \notin U$. $U = \text{names}_i(x)$ implies $x \notin G^i$. On the other hand $E \in V \cup \{\top\}$ and $V = \text{names}_i(y)$ yields $y \in E^i$. Therefore $x \in (\exists r.E)^i$ holds. This shows that $\exists r.E \sqsubseteq G$ does not hold in i, which contradicts the fact that i is a model of $\mathcal{T}$. Hence our assumption that V is not an r-successor of U in κ_R must be false. This proves (S2) for Z_i. We have thus shown that Z_i is a simulation from i to κ_R. □

Minimal Possible Consequences for TBoxes

We still consider the simplified setting where no ABox is present, where the TBox is in normal form and where we are looking for minimal possible consequences with respect to the full set of concept names used in the TBox. Let $\mathcal{T}$ be an $\mathcal{EL}^\bot$-TBox, R a restriction set of A with respect to $\mathcal{T}$ and κ_R the canonical restricted model with respect to $\mathcal{T}$ and $A \to R$. Remember that in Section 4.1.2 we have defined the concept description of R in κ_R to be the concept description that has the same $\mathcal{EL}$-description graph as κ_R and uses R as its root concept. We denote by $C_R = (R, \mathcal{T}_R)$ the concept description of R in κ_R. In this section we

first prove that C_R is a possible consequence of A with respect to $\mathcal{T}$. In a second step we show that if R is maximal among the restriction sets of A with respect to $\mathcal{T}$ then C_R is a minimal possible consequence of A. This proves that minimal possible consequences must exist and gives us an effective method to compute them.

Lemma 7.6. *The canonical restricted model κ_R is a witness model for $A \sqsubseteq C_R$ with respect to $\mathcal{T}$.*

Proof. To prove that κ_R is a witness model for $A \sqsubseteq C_R$ we need to show that $A^{\kappa_R} \subseteq C_R^{\kappa_R}$ holds. Let $U \in \mathcal{U}_{\max}$ be an individual that satisfies $U \in A^{\kappa_R}$, which is equivalent to $A \in U$. To prove that $U \in C_R^{\kappa_R}$ holds, we use Lemma 2.2 and prove that the subset relation $\subseteq$ is a simulation from C_R to U in κ_R. It holds that $\Delta_{\kappa_R} = \mathcal{N}_D(\mathcal{T}_R) = \mathcal{U}_{\max}$ and therefore $\subseteq$ is a relation between the defined concept names of $\mathcal{T}_R$ and the individuals in Δ_{κ_R}. *(S1′)* Let $V, W \in \mathcal{U}_{\max}$ be two sets satisfying $V \subseteq W$. It holds that $\text{names}_{\mathcal{T}_R}(V) = V$ and $\text{names}_{\kappa_R}(W) = W$. Hence V and W satisfy $\text{names}_{\mathcal{T}_R}(V) \subseteq \text{names}_{\kappa_R}(W)$. This proves (S1′) for $\subseteq$.

(S2′) Let $V, W \in \mathcal{U}_{\max}$ be two sets satisfying $V \subseteq W$, let r be a role name and let $V' \in \text{succ}^r_{\mathcal{T}_R}(V)$ be an r-successor of V. Since $\mathcal{T}_R$ and κ_R share the same $\mathcal{EL}$-description graph $V' \in \text{succ}^r_{\mathcal{T}_R}(V)$ is equivalent to $V' \in \text{succ}^r_{\kappa_R}(V)$. From Definition 7.8 it follows that $V' \subseteq \text{can}^r_{\mathcal{T}}(V)$. From $V \subseteq W$ and (7.6) we obtain $\text{can}^r_{\mathcal{T}}(V) \subseteq \text{can}^r_{\mathcal{T}}(W)$ and therefore $V' \subseteq \text{can}^r_{\mathcal{T}}(W)$. Hence Definition 7.8 yields $V' \in \text{succ}^r_{\kappa_R}(W)$. Since obviously $V' \subseteq V'$ holds this proves (S2′) for $\subseteq$.

(S3′) Since κ_R is restricted with respect to $A \to R$ and we know that $A \in U$ it follows that $R \subseteq U$. This proves (S3′) for $\subseteq$.

Hence, $\subseteq$ is a simulation from C_R to U in κ_R. Lemma 2.3 proves that $U \in C_R^{\kappa_R}$ holds. We have thus shown that $A \sqsubseteq C_R$ holds in κ_R, which proves that κ_R is a witness model. □

Corollary 7.7. *C_R is a possible consequence of A with respect to $\mathcal{T}$.*

Knowing that C_R is a possible consequence of A we would like to show that it is also minimal among the possible consequences of A. The following lemma helps us to show this.

Lemma 7.8. *Assume that $\bot$ is not a possible consequence of A with respect to $\mathcal{T}$. If i is a model that is maximally restricted with respect to $A \to R$ then $C_R \sqsubseteq A^{ii}$ holds.*

Proof. Let G_i be the description graph of i. A^i is not empty since $\bot$ is not a possible consequence of A. Let the elements of A^i be enumerated as $A^i = \{a_1, \ldots, a_n\}$. We know from Lemma 4.6 that A^{ii} can be obtained by computing the least common subsumer of the descriptions $\{a_k\}^i$ for all $a_k \in A^i$. According to Lemma 4.5 the model-based most specific concepts $\{x\}^i$ have G_i as their description graph. Lemma 2.4 states that the description graph of their least common subsumer can be obtained as the product $G_i \otimes \cdots \otimes G_i$ where the number of factors is n. The root concept of A^{ii} corresponds to the tuple $(a_1, \ldots, a_n)$ in $G_i \otimes \cdots \otimes G_i$.

If we convert $G_i \otimes \cdots \otimes G_i$ to an interpretation j according to Definition 2.22 then Lemma 5.15 shows that j is a model of $\mathcal{T}$. From Lemma 7.5 we obtain that

$$Z_j = \{(x, \mathrm{names}_j(x)) \mid x \in \Delta_j\}$$

is a simulation from j to κ_R. The models j and κ_R share the same description graph as A^{ii} and C_R, respectively. Therefore, to prove that Z_j is a simulation from A^{ii} to C_R it suffices to show that Z_j contains the pair $\big((a_1, \ldots, a_n), R\big)$, the pair of the respective root concepts. Definition 2.25 yields

$$\mathrm{names}_j\big((a_1, \ldots, a_n)\big) = \bigcap_{k \in \{1,\ldots,n\}} \mathrm{names}_i(a_k) = \bigcap_{a \in A^i} \mathrm{names}_i(a).$$

Since i is maximally restricted with respect to $A \to R$ it holds that

$$\bigcap_{a \in A^i} \mathrm{names}_i(a) = R.$$

This proves $\big((a_1, \ldots, a_n), R\big) \in Z_j$. We have thus shown that Z_j is a simulation from A^{ii} to C_R and thus $C_R \sqsubseteq A^{ii}$ follows from Lemma 2.3. □

Theorem 7.9. *Assume that $\bot$ is not a possible consequence of A with respect to $\mathcal{T}$. Let R be maximal with respect to $\subseteq$ among the restriction sets for A with respect to $\mathcal{T}$. Then C_R is a minimal possible consequence of A with respect to $\mathcal{T}$.*

Proof. Assume that $D = (A_D, \mathcal{T}_D)$ is a possible consequence of A with respect to $\mathcal{T}$ that satisfies $D \sqsubseteq C_R$. Let i be a witness model for $A \sqsubseteq D$.

$D \sqsubseteq C_R \sqsubseteq \bigsqcap R$ yields that the GCI $A \sqsubseteq \bigsqcap R$ must also hold in i. Thus i is restricted with respect to $A \to R$. Since R is maximal among the restriction sets for A with respect to $\mathcal{T}$ it follows that i is also maximally restricted with respect to $A \to R$. We obtain from Lemma 7.8 that $C_R \sqsubseteq A^{ii}$ holds. From $A^i \subseteq D^i$ and Lemma 4.1 we obtain that $A^{ii} \sqsubseteq D^{ii} \sqsubseteq D$ holds. This proves $C_R \sqsubseteq D$ and therefore C_R is a minimal possible consequence for A with respect to $\mathcal{T}$. □

Corollary 7.10. *If $\mathcal{T}$ is an $\mathcal{EL}^\bot$-TBox in normal form over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ and $A \in \mathcal{N}_C$ is a concept name then $\mathfrak{mpc}_\mathcal{T}^{\mathcal{N}_C}(A)$ is not empty, i. e. minimal possible consequences exist.*

Minimal Possible Consequences for Ontologies

In the previous sections we have shown that minimal possible consequences exist in a setting where only a TBox is present. In this section we allow an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ that consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. That is we show that $\mathfrak{mpc}_\mathcal{O}^{\mathcal{N}C}(A)$ is not empty, where

- $A \in \mathcal{N}_C$ is a concept name,
- $\mathcal{O}$ is an ontology over the signature $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$, and
- $\mathcal{T}$ is a general $\mathcal{EL}^\bot$-TBox in its normal form (7.4).

The prove closely ressembles the proof for TBoxes.

Definition 7.9 (Restriction Set)**.** R is called a *restriction set* of A with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ if there is a model i of $\mathcal{O}$ such that i is maximally restricted with respect to $A \to R$.

If R is a restriction set of A with respect to $\mathcal{O}$ and i is a model of $\mathcal{O}$ that is restricted with respect to $A \to R$ then we can extend the canonical model κ_R from Definition 7.8 by defining

$$a^{\kappa_R} = \mathrm{names}_i(a^i) \tag{7.7}$$

for all individual names $a \in \mathcal{N}_I$.

Lemma 7.11. *Let R be a restriction set of A with respect to $\mathcal{O}$ and let i be a model of $\mathcal{O}$ that is restricted with respect to $A \to R$. Then κ_R extended according to (7.7) is a witness model for $A \sqsubseteq C_R$ with respect to $\mathcal{O}$.*

Proof. We already know that $A \sqsubseteq C_R$ holds in κ_R and that κ_R is a model of $\mathcal{T}$ from Lemma 7.6. Thus to prove that κ_R is a witness model it only remains to show that κ_R is also a model of $\mathcal{A}$. We look at concept assertions and role assertions separately. *Concept Assertions:* Let $B(a)$, where $B \in \mathcal{N}_C$ and $a \in \mathcal{N}_I$, be a concept assertion in $\mathcal{A}$. Since i is a model of $\mathcal{A}$ it follows that $a^i \in B^i$ and therefore $B \in \text{names}_i(a^i) = a^{\kappa_R}$ holds. Definition 7.8 yields $a^{\kappa_R} \in B^{\kappa_R}$. Therefore $B(a)$ holds in κ_R. *Role Assertions:* Let $r(a, b)$, where $r \in \mathcal{N}_R$ and $a, b \in \mathcal{N}_I$, be a role assertion in $\mathcal{A}$. Since i is a model of $\mathcal{A}$ it holds that $b^i \in \text{succ}_i^r(a^i)$. Because i is a model of $\mathcal{T}$ this implies $\text{names}_i(b^i) \subseteq \text{can}_{\mathcal{T}}^r(\text{names}_i(a^i))$, which is equivalent to $b^{\kappa_R} \subseteq \text{can}_{\mathcal{T}}^r(a^{\kappa_R})$. Definition 7.8 yields $(a^{\kappa_R}, b^{\kappa_R}) \in r^{\kappa_R}$, i. e. $r(a, b)$ holds in κ_R. This proves that all statements from $\mathcal{A}$ hold in κ_R. Thus κ_R is a witness model of $A \sqsubseteq C_R$ with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$. □

Corollary 7.12. *If R is a restriction set of A with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, then C_R is a possible consequence of A with respect to $\mathcal{O}$.*

Our argument to prove minimality of C_R is the same as for the case where only a TBox is present.

Theorem 7.13. *Assume that $\bot$ is not a possible consequence of A with respect to $\mathcal{O}$. If R is maximal with respect to $\subseteq$ among the restriction sets for A with respect to $\mathcal{O}$ then C_R is a minimal possible consequence for A with respect to $\mathcal{T}$.*

Proof. Assume that D is a possible consequence of A with respect to $\mathcal{O}$ and that D satisfies $D \sqsubseteq C_R$. Let j be a witness model for $A \sqsubseteq D$ with respect to $\mathcal{O}$. Using the same argument as in the proof of Theorem 7.9 we can show that j is maximally restricted with respect to $A \to R$. Lemma 7.8 then yields $C_R \sqsubseteq A^{jj}$. From $A^j \subseteq D^j$ and Lemma 4.1 we obtain $A^{jj} \sqsubseteq D^{jj} \sqsubseteq D$ and thus $C_R \sqsubseteq D$. This proves that C_R is a minimal possible consequence of A with respect to $\mathcal{O}$. □

We have thus proven minimality of C_R as a possible consequence of A, provided that R is maximal among the restriction sets. We can prove an even stronger result, namely that for every possible consequence D there is some maximal restriction set R such that C_R is more specific than D.

Lemma 7.14. *Assume that $\bot$ is not a possible consequence of A with respect to $\mathcal{O}$. If D is a possible consequence of A with respect to $\mathcal{O}$ then there is some R such that $C_R \sqsubseteq D$.*

Proof. Let i be a witness model for $A \sqsubseteq D$ with respect to $\mathcal{O}$. Let $R \subseteq \mathcal{N}_C$ be the set of concept names for which i is maximally restricted with respect to $A \to R$. Then we obtain $C_R \sqsubseteq A^{ii}$ from Lemma 7.8 and using Lemma 4.1 we obtain $C_R \sqsubseteq A^{ii} \sqsubseteq D^{ii} \sqsubseteq D$. □

Corollary 7.15. *Assume that $\bot$ is not a possible consequence of A with respect to $\mathcal{O}$. If D is a possible consequence of A with respect to $\mathcal{O}$ then there is some R such that*

- *C_R is a minimal possible consequence of A with respect to $\mathcal{O}$, and*
- *$C_R \sqsubseteq D$ holds.*

Proof. This is readily obtained from Lemma 7.14 and the fact that the set $\{C_R \mid R \text{ is a restriction set for } A \text{ with respect to } \mathcal{O}\}$ is finite. □

The last result is even stronger than a simple existence result for minimal possible consequences. It shows that not only do minimal possible consequences exist, but that every possible consequence D subsumes a minimal possible consequence.

Minimal Possible Consequences for Restricted Signatures

In this section we successively remove the remaining restrictions. We start with the restriction on the signature. We assume that $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is an ontology that may use an extended signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$. We show that $\mathtt{mpc}_{\mathcal{O}}^{\mathcal{N}_C}(A)$, where $A \in \mathcal{N}_C$ is a concept name, is not empty. We still assume that $\mathcal{T}$ is in its normal form. As in the previous sections the case where $\bot$ is a possible consequence is trivial. We assume for the rest of this section that $\bot$ is not a possible consequence of the concept in question.

When $C = (A_C, \mathcal{T}_C)$ is an $\mathcal{EL}_{\text{gfp}}$-concept description that uses the defined concept names from $\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}$, we define the concept description $C|_{\mathcal{N}_C}$ to be the restriction of C to the concept names from $\mathcal{N}_C$, i. e. the concept description $C|_{\mathcal{N}_C} = (A_C, \mathcal{T}_C|_{\mathcal{N}_C})$ where for every statement

$$D \equiv \bigsqcap P \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E$$

from $\mathcal{T}_C$, where $D \in \mathcal{N}_D(\mathcal{T}_C)$, $P \subseteq \mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}$, and $\Pi \subseteq \mathcal{N}_R \times \mathcal{N}_D(\mathcal{T}_C)$, the TBox $\mathcal{T}_C|_{\mathcal{N}_C}$ contains the statement

$$D \equiv \bigsqcap (P \cap \mathcal{N}_C) \sqcap \bigsqcap_{(r,E)\in\Pi} \exists r.E.$$

The $\mathcal{EL}$-description graph of $C|_{\mathcal{N}_C}$ is obtained from the $\mathcal{EL}$-description graph of C by removing all concept names from $\mathcal{N}_C^{\text{new}}$. Let $\bar{C}$ be an $\mathcal{EL}_{\text{gfp}}$-concept description over the smaller signature $(\mathcal{N}_C, \mathcal{N}_R)$. If Z is a simulation from $\bar{C}$ to C then Z is also a simulation from $\bar{C}$ to $C|_{\mathcal{N}_C}$, because Z still satisfies (S1″) since no concept names from $\mathcal{N}_C^{\text{new}}$ occur in $\bar{C}$. (S2″) and (S3″) remain unaffected if concept names are removed from the description graph. Therefore, if $\bar{C}$ uses only concept names from $\mathcal{N}_C$ then

$$C \sqsubseteq \bar{C} \text{ implies } C|_{\mathcal{N}_C} \sqsubseteq \bar{C}. \tag{7.8}$$

Lemma 7.16. *If D is an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ and D is a possible consequence of A with respect to $\mathcal{O}$ then there is a restriction set $R \subseteq \mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}$ such that $C_R|_{\mathcal{N}_C} \sqsubseteq D$.*

Proof. The claim follows immediately from Lemma 7.14 and the fact that $C_R \sqsubseteq D$ implies $C_R|_{\mathcal{N}_C} \sqsubseteq D$. □

Corollary 7.17. *Let C be minimal with respect to $\sqsubseteq$ within the set*

$$\mathcal{X} = \{C_R|_{\mathcal{N}_C} \mid R \subseteq \mathcal{N}_C \cup \mathcal{N}_C^{\text{new}} \text{ is a restriction set of } A \text{ w.r.t. } \mathcal{O}\}.$$

Then C is a minimal possible consequence of A with respect to $\mathcal{O}$.

Proof. Let D be a possible consequence of A with respect to $\mathcal{O}$ that satisfies $D \sqsubseteq C$. Then Lemma 7.16 implies that there is some $E \in \mathcal{X}$ satisfying $E \sqsubseteq D$ and thus also $E \sqsubseteq C$. Minimality of C proves $E \equiv D \equiv C$. □

The second restriction that we need to remove is the restriction that the TBox $\mathcal{T}$ needs to be in normal form. In [BBL05b] it has been shown that any general $\mathcal{EL}^{\bot}$-TBox can be converted into this normal form in linear time. After the conversion one obtains a new 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 $\mathcal{T}'$ is a representation of every model of $\mathcal{T}$. The conversion introduces

new concept names.[3] Let D be a concept description that uses only concept names that occur in $\mathcal{T}$. With $\mathcal{T}'$ being a conservative extension of $\mathcal{T}$ we obtain that if D is a possible consequence of A with respect to $\mathcal{O}' = (\mathcal{T}', \mathcal{A})$, then it is also a possible consequence with respect to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ and vice versa. Since $\mathcal{T}'$ is in normal form Corollary 7.17 proves existence of minimal possible consequences for both cases.

The last restriction that we can only compute minimal possible consequences for concept names, and not for complex concept descriptions, is relatively easy to remove. Let C be an acyclic $\mathcal{EL}^{\perp}_{\text{gfp}}$-concept description. We can simply add a new concept name A_C and a TBox statement $A_C \equiv C$ before doing the conversion to normal form. Clearly, since minimal possible consequences for A_C exist, they must also exist for C. We do not consider the case where C is cyclic.

Theorem 7.18. *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}^{\perp}$-ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$, and let C be an acyclic concept description over the signature $\mathcal{N}_C$. Then there is a minimal possible consequence of C with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$.*

The above existence proofs also yield a construction for minimal possible consequences, consisting of the following steps

1. Add a statement $A_C \equiv C$ to $\mathcal{T}$.
2. Convert $\mathcal{T}$ to its normal form $\mathcal{T}'$.
3. Using a reasoner and consistency checking determine all restriction sets of A_C with respect to $(\mathcal{T}', \mathcal{A})$.
4. Compute the set of all concept descriptions C_R for all restriction sets R and find its minimal elements.

Clearly, while this is an effective procedure it is not efficient since the latter two steps require exponential time and space. It remains open whether efficient algorithms can be obtained. Instead of focussing on efficient algorithm for computing minimal possible consequences we propose to use approximated possible consequences, which will be introduced in the following section.

[3] The conversion presented in [BBL05b] uses a more expressive variant of $\mathcal{EL}$ that allows for role inclusions. Therefore, the conversion presented there can also add new role names. Since we do not allow role inclusions in $\mathcal{EL}^{\perp}$ no new role names are added in our setting.

7.2.3 Approximation of Minimal Possible Consequences

Minimal possible consequences have several unpleasant properties, e. g. they are not always unique. In the previous section we have introduced a naive algorithm for computing them. This algorithm involves computing the power set of $\mathcal{N}_C$, thus it requires exponential time and space. In practice, it can only be used for very small sets $\mathcal{N}_C$. To avoid the difficulties associated with minimal possible consequences we propose two simplifications. First, we define approximations of minimal possible consequences that are easier to compute. Second, we restrict the expressivity of the ontology by only allowing statements of certain types, namely the types used in (7.1) and (7.2).

The fact that counterexamples need not be explicit is a contributing factor to the difficulties when computing possible consequences. It is essentially the reason why we have to use restricted canonical models instead of canonical models in Section 7.2.2. The idea in the following is to approximate possible consequences by simply ignoring non-explicit counterexamples.

Definition 7.10 (Approximated Possible Consequences)**.** Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ an ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$. D is called an *approximated possible consequence* of C with respect to $\mathcal{O}$ if there is no individual $a \in \mathcal{N}_I$ that is an explicit counterexample to $C \sqsubseteq D$ with respect to $\mathcal{O}$. We denote by $\mathfrak{apc}_{\mathcal{O}}^{\mathcal{N}_C}(C)$ the set of all $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ that are approximated possible consequences of C with respect to $\mathcal{O}$. D is called *minimal approximated possible consequence* of C with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$ if D is minimal within the set $\mathfrak{apc}_{\mathcal{O}}^{\mathcal{N}_C}(C)$ with respect to $\sqsubseteq$.

Lemma 7.19. *If D is a possible consequence of C with respect to $\mathcal{O}$ then D is an approximated possible consequence of C with respect to $\mathcal{O}$.*

Proof. Let $a \in \mathcal{N}_I$ be an individual that satisfies $\mathcal{O} \models C(a)$. Since D is a possible consequence of C there exists a witness model i for $C \sqsubseteq D$. It holds that $a^i \in C^i \subseteq D^i$. We have thus disproved $\mathcal{O} \models \neg D(a)$. Hence a is not an explicit counterexample for the GCI $C \sqsubseteq D$. Because a was an arbitrary individual name this shows that there cannot be an explicit counterexample to $C \sqsubseteq D$ and D must be an approximated possible consequence. □

Lemma 7.19 shows in particular that a possible consequence cannot be strictly more specific than a minimal approximated possible consequence. Approximated possible consequences are closely related to the following notion.

Definition 7.11 (Possible Description)**.** Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$ and $a \in \mathcal{N}_I$ an individual. D is called a *possible description* of a with respect to $\mathcal{O}$ if $\mathcal{O} \not\models \neg D(a)$, i. e. if there is a model i of $\mathcal{O}$ such that $a^i \in D^i$ holds. We denote by $\mathfrak{pd}_{\mathcal{O}}^{\mathcal{N}_C}(a)$ the set of all $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ that are possible descriptions of a with respect to $\mathcal{O}$.

D is called *minimal possible description* of a with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$ if D is minimal within the set $\mathfrak{pd}_{\mathcal{O}}^{\mathcal{N}_C}(C)$ with respect to $\sqsubseteq$.

In Section 7.1.2 we have seen that a background model i can be completely described using only the TBox statements from (7.1) and the ABox statement from (7.2). In the rest of this section we look at TBoxes and ABoxes that are subsets of the TBoxes and ABoxes from (7.1) and (7.2). That is, we assume that $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is an ontology over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I')$, where $\mathcal{N}_C^{\text{new}} = \{T_a, F_a \mid a \in \mathcal{N}_I'\}$, $\mathcal{N}_C$ and $\mathcal{N}_C^{\text{new}}$ are disjoint and $\mathcal{N}_I' = \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}}$. We require that $\mathcal{A}$ contains the concept assertion

$$T_a(a) \tag{7.9}$$

for all individuals $a \in \mathcal{N}_I'$. Optionally, $\mathcal{A}$ may contain concept assertions of the form

$$A(a), \tag{7.10}$$

where $A \in \mathcal{N}_C$ and $a \in \mathcal{N}_I'$, and $\mathcal{A}$ may contain role assertions

$$r(a, b), \tag{7.11}$$

where $a, b \in \mathcal{N}_I'$ and $r \in \mathcal{N}_R$. Similarly, we require that $\mathcal{T}$ contains the GCI

$$T_a \sqcap F_a \sqsubseteq \bot \tag{7.12}$$

for every individual $a \in \mathcal{N}_I'$. Optionally, $\mathcal{T}$ may contain statements of the form

$$A \sqsubseteq F_a, \tag{7.13}$$

where $A \in \mathcal{N}_C$ and $a \in \mathcal{N}_I'$. For every pair $(a, r) \in \mathcal{N}_I' \times \mathcal{N}_R$ the TBox $\mathcal{T}$ may contain at most one statement of the form

$$\exists r.(\bigsqcap P) \sqsubseteq F_a, \tag{7.14}$$

where $P \subseteq \{F_b \mid b \in \mathcal{N}_I'\}$, and where $r(a, b) \in \mathcal{A}$ implies $F_b \in P$. Otherwise, we allow P to be empty and define $\bigsqcap \emptyset = \top$ as usual. Furthermore, we assume that $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is consistent. The intuition behind statements of type (7.13) is that they can specify to which concept names an individual does not belong. Statements of type (7.14) specify which r-successors an individual can have.

In the previous sections we have constructed canonical models for $\mathcal{EL}^\bot$-ontologies with general TBoxes. We construct an interpretation $\kappa_\mathcal{O} = (\Delta_{\kappa_\mathcal{O}}, \cdot^{\kappa_\mathcal{O}})$ of $\mathcal{O}$ that serves a similar purpose. We define

$$\Delta_{\kappa_\mathcal{O}} = \mathcal{N}_I' \cup \{\alpha\},$$

where $\alpha \notin \mathcal{N}_I'$ is a new individual name. We define the interpretation function $\cdot^{\kappa_\mathcal{O}}$ as follows. For every individual name $a \in \mathcal{N}_I'$ we define

$$a^{\kappa_\mathcal{O}} = a.$$

We define

$$A^{\kappa_\mathcal{O}} = \{\alpha\} \cup \{a \mid A \sqsubseteq F_a \notin \mathcal{T}\}, \quad T_a^{\kappa_\mathcal{O}} = \{a\}, \quad F_a^{\kappa_\mathcal{O}} = \Delta_{\kappa_\mathcal{O}} \setminus \{a\} \tag{7.15}$$

for all concept names $A \in \mathcal{N}_C$ and all individual names $a \in \mathcal{N}_I'$. Lastly, for every role name $r \in \mathcal{N}_R$ we define

$$\begin{aligned} r^{\kappa_\mathcal{O}} = &\{(\alpha, \alpha)\} \\ &\cup \{(a, b) \mid \exists r.\bigsqcap P \sqsubseteq F_a \in \mathcal{T} \text{ implies } F_b \in P\} \\ &\cup \{(a, \alpha) \mid \text{there is no GCI } \exists r.\bigsqcap P \sqsubseteq F_a \in \mathcal{T}\}. \end{aligned} \tag{7.16}$$

Lemma 7.20. *If $\mathcal{O}$ is consistent then the interpretation $\kappa_\mathcal{O}$ is a model of $\mathcal{O}$.*

Proof. We verify for each statement from $\mathcal{O}$ that it holds in $\kappa_\mathcal{O}$. *ABox-assertions:* We have defined $T_a^{\kappa_\mathcal{O}} = \{a\}$ and thus all assertions of the form $T_a(a) \in \mathcal{A}$ hold in $\kappa_\mathcal{O}$. Let $A(a) \in \mathcal{A}$ be an assertion of type (7.10). We know that $\mathcal{T}$ contains the GCI $T_a \sqcap F_a \sqsubseteq \bot$ and $\mathcal{A}$ contains $T_a(a)$. Since $\mathcal{A}$ also contains $A(a)$ there cannot be a GCI $A \sqsubseteq F_a$ in $\mathcal{T}$, otherwise $\mathcal{O}$ would be inconsistent. Thus $a \in A^{\kappa_\mathcal{O}}$ follows from (7.15). Therefore $A(a)$ holds in $\kappa_\mathcal{O}$. Assume that $\mathcal{A}$ contains $r(a,b)$. We distinguish two cases. If there is a statement $\exists r.(\bigsqcap P) \sqsubseteq F_a$ in $\mathcal{T}$ then $r(a,b) \in \mathcal{A}$ implies that $F_b \in P$ holds. Then (7.16) yields $(a,b) \in r^{\kappa_\mathcal{O}}$. If there is no statement $\exists r.(\bigsqcap P) \sqsubseteq F_b$ then $(a,b) \in r^{\kappa_\mathcal{O}}$ holds according to (7.16). We have thus shown that all ABox-assertions from $\mathcal{A}$ hold in $\kappa_\mathcal{O}$.

TBox: Let $T_a \sqcap F_a \sqsubseteq \bot$ be a GCI of type (7.12) in $\mathcal{T}$. By definition it holds that $T_a^{\kappa_\mathcal{O}} \cap F_a^{\kappa_\mathcal{O}} = \{a\} \cap (\Delta_{\kappa_\mathcal{O}} \setminus \{a\}) = \emptyset$. Thus $T_a \sqcap F_a \sqsubseteq \bot$ holds in $\kappa_\mathcal{O}$. Let $A \sqsubseteq F_a$ be a GCI of type (7.13). The definition of $A^{\kappa_\mathcal{O}}$ yields $a \notin A^{\kappa_\mathcal{O}}$. Thus we obtain $A^{\kappa_\mathcal{O}} \subseteq \Delta_{\kappa_\mathcal{O}} \setminus \{a\} = F_a^{\kappa_\mathcal{O}}$. Thus $A \sqsubseteq F_a$ holds in $\kappa_\mathcal{O}$.

Lastly, let $\exists r.(\bigsqcap P) \sqsubseteq F_a$ be a GCI of type (7.14) in $\mathcal{T}$. Then (7.16) yields $(a,\alpha) \notin r^{\kappa_\mathcal{O}}$, i. e. α is not an r-successor of a. Let $b \in \mathcal{N}_I'$ be an individual name satisfying $(a,b) \in r^{\kappa_\mathcal{O}}$. (7.16) yields $F_b \in P$. On the other hand $b \notin F_b^{\kappa_\mathcal{O}} = \Delta_{\kappa_\mathcal{O}} \setminus \{b\}$ holds, which implies $b \notin (\bigsqcap P)^{\kappa_\mathcal{O}}$. Hence, no r-successor of a is in $(\bigsqcap P)^{\kappa_\mathcal{O}}$, which implies that $a \notin (\exists r.\bigsqcap P)^{\kappa_\mathcal{O}}$ holds. We obtain $(\exists r.\bigsqcap P)^{\kappa_\mathcal{O}} \subseteq \Delta_{\kappa_\mathcal{O}} \setminus \{a\} = F_a^{\kappa_\mathcal{O}}$. Thus $\exists r.\bigsqcap P \sqsubseteq F_a$ holds in $\kappa_\mathcal{O}$. □

We denote by $C_a = (a, \mathcal{T}_a)$ the concept description of a in $\kappa_\mathcal{O}$. Clearly, $a^{\kappa_\mathcal{O}} \in C_a^{\kappa_\mathcal{O}}$ holds ($\kappa_\mathcal{O}$ and C_a share the same description graph and thus the identity relation serves as a simulation). Lemma 7.20 then shows that C_a is a possible description of a with respect to $\mathcal{O}$. Furthermore, the restriction $C_a|_{\mathcal{N}_C}$ to the set of concept names $\mathcal{N}_C$ is also a possible description of a with respect to $\mathcal{O}$.

Lemma 7.21. *Let $D \in \mathfrak{pd}_\mathcal{O}^{\mathcal{N}_C}(a)$ be a possible description of $a \in \mathcal{N}_I'$ for the set of concept names $\mathcal{N}_C$. Then it holds that $C_a|_{\mathcal{N}_C} \sqsubseteq D$.*

This proves in particular that $C_a|_{\mathcal{N}_C}$ is the least element of $\mathfrak{pd}_\mathcal{O}^{\mathcal{N}_C}(a)$ and thus also a minimal possible description of a. Since $\mathfrak{pd}_\mathcal{O}^{\mathcal{N}_C}(a)$ contains a least element minimal possible descriptions of a must be unique up to equivalence.

Proof. We want to prove $C_a|_{\mathcal{N}_C} \sqsubseteq D$. The description $\bot$ cannot be a possible description of a and therefore D must be an $\mathcal{EL}_{\text{gfp}}$-concept description $D = (A_D, \mathcal{T}_D)$. Since D uses only the concept names from $\mathcal{N}_C$ it suffices according to (7.8) to prove $C_a \sqsubseteq D$. By Lemma 2.3 we need to show that there is a simulation from D to C_a. C_a and $\kappa_\mathcal{O}$ share the same $\mathcal{EL}$-description graph. Thus it suffices to show that there is a simulation from D to a in $\kappa_\mathcal{O}$.

Let $i = (\Delta_i, \cdot^i)$ be a model of $\mathcal{O}$ that satisfies $a \in D^i$. We define the relation

$$Z_2 = \{(x, b) \in \Delta_i \times \mathcal{N}_I' \mid x \notin F_b^i\} \cup \{(x, \alpha) \mid x \in \Delta_i\}.$$

While Z_2 is not necessesarily a simulation we show that $\text{names}_i(x) \cap \mathcal{N}_C \subseteq \text{names}_{\kappa_\mathcal{O}}(y)$ for all pairs $(x, y) \in Z_2$, i. e. Z_2 satisfies (S1) for the original concept names from $\mathcal{N}_C$, but not necessarily for the names from $\mathcal{N}_C^{\text{new}}$. Furthermore, we show that Z_2 satisfies (S2). *All pairs* $(x, y) \in Z_2$ *satisfy* $\text{names}_i(x) \cap \mathcal{N}_C \subseteq \text{names}_{\kappa_\mathcal{O}}(y)$*:* Let $(x, b) \in Z_2$ be a pair in the relation Z_2, where $b \neq \alpha$. This yields $x \notin F_b^i$. Let $A \in \text{names}_i(x) \cap \mathcal{N}_C$ be a concept name. Then $x \in A^i$ must hold. Then $x \in A^i$ and $x \notin F_b^i$ yield that $A \sqsubseteq F_b$ does not occur in $\mathcal{T}$, for otherwise i would not be a model of $\mathcal{T}$. The definition of $\kappa_\mathcal{O}$ implies that $b \in A^{\kappa_\mathcal{O}}$ and therefore $A \in \text{names}_{\kappa_\mathcal{O}}(b)$. For pairs $(x, \alpha) \in Z_2$ the claim is trivial since $\mathcal{N}_C \subseteq \text{names}_{\kappa_\mathcal{O}}(\alpha)$.

(S2) Let $(x, b) \in Z_2$ be a pair in the relation Z_2, where $b \neq \alpha$. Let $y \in \text{succ}_i^r(x)$ be an r-successor of x in i. If $\mathcal{T}$ does not contain a statement of the form $\exists r.\prod P \sqsubseteq F_b$ then it holds that $(b, \alpha) \in r^{\kappa_\mathcal{O}}$. Furthermore, $(y, \alpha) \in Z_2$ holds, which proves (S2). In the case where $\mathcal{T}$ does contain a statement of the form $\exists r.\prod P \sqsubseteq F_b$ we know that $y \notin (\prod P)^i$ holds. Hence, there must be some concept name $F_c \in P$ satisfying $y \notin F_c^i$ and thus $(y, c) \in Z_2$ holds. The definition of $\kappa_\mathcal{O}$ yields $(b, c) \in r^{\kappa_\mathcal{O}}$. This proves (S2).

Since $i = (\Delta_i, \cdot^i)$ satisfies $a \in D^i$, Lemma 2.2 yields that there is a simulation Z_1 from D to a^i in i. Since D uses only the concept names from $\mathcal{N}_C$ the first result suffices to prove that $Z_2 \circ Z_1$ satisfies (S1), while the second result implies that $Z_2 \circ Z_1$ satisfies (S2). It is easy to verify that $Z_2 \circ Z_1$ contains the pair (A_D, a). Thus $Z_2 \circ Z_1$ is a simulation from D to a in $\kappa_\mathcal{O}$ and thus also a simulation from D to C_a. Lemma 2.3 yields $C_a \sqsubseteq D$, and therefore $C_a|_{\mathcal{N}_C} \sqsubseteq D$ must also be true. □

Theorem 7.22. *Let $\mathcal{O}$ be an ontology that only contains statements of the types (7.9), (7.10), (7.11), (7.12), (7.13), and (7.14). Let C and D be $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions. D is an approximated possible consequence of C with respect to $\mathcal{O}$ iff $C_a|_{\mathcal{N}_C} \sqsubseteq D$ holds for all individuals $a \in \mathcal{N}_I'$ that satisfy $\mathcal{O} \models C(a)$.*

Proof. D is not an approximated possible consequence of C iff there is an explicit counterexample $a \in \mathcal{N}_I'$ to $C \sqsubseteq D$. An individual $a \in \mathcal{N}_I'$ is an explicit counterexample to $C \sqsubseteq D$ iff a satisfies $\mathcal{O} \models C(a)$ and $\mathcal{O} \models \neg D(a)$. Furthermore, a satisfies $\mathcal{O} \models \neg D(a)$ iff there is no model i in which $a^i \in D^i$ holds, i. e. iff D is not a possible description of a. Lemma 7.21 states that D is not a possible description of a if $C_a|_{\mathcal{N}_C} \not\sqsubseteq D$ holds. On the other hand, if $C_a|_{\mathcal{N}_C} \sqsubseteq D$ holds then $a \in D^{\kappa_{\mathcal{O}}}$ must also hold, since C_a and $\kappa_{\mathcal{O}}$ share the same $\mathcal{EL}$-description graph. This shows that D is not a possible description of a if *and only if* $C_a|_{\mathcal{N}_C} \not\sqsubseteq D$ holds.

In summary, we obtain that D is not an approximated possible consequence of C iff there is an individual $a \in \mathcal{N}_I'$ that satisfies $\mathcal{O} \models C(a)$ and $C_a|_{\mathcal{N}_C} \not\sqsubseteq D$. □

Corollary 7.23. *Let E be the least common subsumer of all descriptions C_a where $a \in \mathcal{N}_I'$ satisfies $\mathcal{O} \models C(a)$. Then E is a minimal approximated possible consequence of C and minimal approximated possible consequences are unique up to equivalence.*

Proof. E subsumes all descriptions C_a for all $a \in \mathcal{N}_I'$ that satisfy $\mathcal{O} \models C(a)$. Theorem 7.22 yields that E is an approximated possible consequence of C. Let D be an approximated possible consequence of C. Then Theorem 7.22 yields that D is a common subsumer of all concept descriptions C_a where $a \in \mathcal{N}_I'$ satisfies $\mathcal{O} \models C(a)$. Hence it follows that $E \sqsubseteq D$. This shows that E is the only minimal approximated possible consequence of C up to equivalence. □

The fact that approximated possible consequence are unique up to equivalence justifies that we can speak of *the* approximated possible consequence of a concept description C with respect to $\mathcal{O}$ for the set of concept names $\mathcal{N}_C$, which we denote by $\mathtt{mapc}_{\mathcal{O}}^{\mathcal{N}_C}(C)$. The following result is readily obtained from Lemma 7.19, which states that possible consequences are always approximated possible consequences, and Corollary 7.23.

Corollary 7.24. *Let* $E = \mathfrak{mapc}_{\mathcal{O}}^{\mathcal{N}_C}(C)$ *be the approximated minimal possible consequence of* C *with respect to* $\mathcal{O}$ *for the set of concept names* $\mathcal{N}_C$*. If* D *is a possible consequence of* C *with respect to* $\mathcal{O}$ *for the set of concept names* $\mathcal{N}_C$ *then it holds that* $E \sqsubseteq D$*.*

In this section we have shown that in the simple case where the ontology contains only statements of certain types, minimal approximated possible consequences are unique up to equivalence. In comparison to minimal possible consequences they can be computed fairly easily, since constructing $\kappa_{\mathcal{O}}$ can be done in linear time in the size of $\mathcal{O}$. Conversely, canonical restricted models have exponential size in the size of the ontology.

7.3 ABox Exploration

The goals of ABox Exploration are essentially the same as the goals of Model-Exploration. The difference is that ABox Exploration uses an ontology to keep track of counterexamples. We present two ontology completion formalisms where counterexamples are stored in an ontology. We call these formalisms ABox Exploration. The two algorithms mainly differ in the way right hand sides of the GCIs are obtained. Algorithm 12 uses minimal possible consequences, while Algorithm 13 uses minimal approximated possible consequences.

For the first algorithm we look at two settings, one where no additional restrictions are placed on the ontology, and one where only statements of types (7.9), (7.10), (7.11), (7.12), (7.13), and (7.14) are allowed.

7.3.1 Exploration Using Minimal Possible Consequences

ABox Exploration, like Model Exploration, assumes that we have access to an expert's complete knowledge about the domain of the knowledge base, and that this knowledge can be represented in the form of a background model i over some signature $(\mathcal{N}_C, \mathcal{N}_R)$. However, i is initially not known to the algorithm. The algorithm only has access to a knowledge base $\mathcal{O}_0$, which is a representation of i. The exploration follows the usual pattern: Questions in the form of GCIs are presented to the

expert who either accepts or refutes them. When a GCI is refuted the expert must extend the current knowledge base $\mathcal{O}_l$ to a new knowledge base $\mathcal{O}_{l+1}$ that refutes the GCI. Naturally, the new knowledge base is required to be a representation of i.

Algorithm 12 is a modification of Algorithm 10, the algorithm for computing a finite base with acyclic left-hand sides for the GCIs holding in a given model. Like in Model Exploration (Algorithm 11) a while-loop is added and the expert interaction takes place within that inner while-loop. For a given left-hand side $\bigsqcap \tilde{P}_k$ the algorithm chooses as the corresponding right-hand side one of the minimal possible consequence of $\bigsqcap \tilde{P}_k$ with respect to $\mathcal{O}_l$. This is possible since $\bigsqcap \tilde{P}_k$ is an acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and we have seen in Section 7.2.2 that minimal possible consequences for acyclic $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description exist and can be computed effectively.

Lemma 7.25. *Let $\mathcal{O}$ be a representation of $i = (\Delta_i, \cdot^i)$ over the signature $(\mathcal{N}_C \cup \mathcal{N}_C^{\text{new}}, \mathcal{N}_R, \mathcal{N}_I \cup \mathcal{N}_I^{\text{new}})$. Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description over the signature $(\mathcal{N}_C, \mathcal{N}_R)$ and $D \in \mathfrak{mpc}_{\mathcal{O}}^{\mathcal{N}_C}(C)$ a minimal possible consequence of C with respect to $\mathcal{O}$. If $C \sqsubseteq D$ holds in i then D satisfies $D \equiv C^{ii}$.*

Proof. Since $C \sqsubseteq D$ holds in i we obtain $C^i \subseteq D^i$. Lemma 4.1 yields $C^{ii} \sqsubseteq D$. On the other hand, i is a witness model for $C \sqsubseteq C^{ii}$, because Lemma 4.1 yields $C^i \subseteq C^{iii}$. Therefore C^{ii} is a possible consequence of C. $C^{ii} \sqsubseteq D$ and minimality of D yield $D \equiv C^{ii}$. □

This shows that, like in Model Exploration, once the expert accepts a GCI $C \sqsubseteq D$ we have found the model-based most specific concept C^{ii}, even though we do not know the background model i explicitly.

In Line 8 of Algorithm 10 a new set of implications $\bar{\mathcal{L}}_{k+1}$ is computed. The right-hand sides of these implications are obtained as the closures of the respective left-hand sides in the induced context $\mathbb{K}_{k+1}$. In the setting of Algorithm 12 the model i is not known and therefore we cannot compute an induced context, let alone the corresponding closure operator. Instead we use projections of the minimal possible consequence D to the current set of attributes M_{k+1}.

Lemma 7.26. *Let M be a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions and let $\mathbb{K}$ be the context induced by i and M. Let $\mathcal{O}$ be a representation of*

Algorithm 12 ABox Exploration

input $\mathcal{O}_0 := (\mathcal{T}_0, \mathcal{A}_0)$
$\tilde{P}_0 := \emptyset$, $\tilde{M}_0 := \mathcal{N}_P \cup \{\bot\}$, $k := 0$, $l := 0$,
$\tilde{\mathcal{S}}_0 := \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_P\}$, $\tilde{\mathcal{L}}_0 := \emptyset$
while $P_k \neq \mathsf{null}$ **do**
  Select $D_k \in \mathfrak{mpc}^{\mathcal{N}_C}_{\mathcal{O}_l}(\bigsqcap \tilde{P}_k)$
  while Expert refutes $\bigsqcap \tilde{P}_k \sqsubseteq D_k$ **do**
    Ask the expert for a new ontology $\mathcal{O}_{l+1}$ that
    – extends $\mathcal{O}_l$,
    – is a representation of the background model, and
    – refutes $\bigsqcap \tilde{P}_k \sqsubseteq D_k$
    $l := l + 1$
    Select new $D_k \in \mathfrak{mpc}^{\mathcal{N}_C}_{\mathcal{O}_l}(\bigsqcap \tilde{P}_k)$
  end while
  $\tilde{M}_{k+1} := \tilde{M}_k \cup \begin{cases} \{\exists r. \bigsqcap \tilde{P}_k \mid r \in \mathcal{N}_R\} & \text{if } D_\ell \not\equiv D_k \text{ for all } \ell < k \\ \emptyset & \text{otherwise} \end{cases}$
  $\tilde{\mathcal{S}}_{k+1} := \{\{A\} \to \{B\} \mid A, B \in \tilde{M}_{k+1}, A \sqsubseteq B\}$
  $\tilde{\mathcal{L}}_{k+1} := \{\tilde{P}_r \to \mathrm{pr}_{\tilde{M}_{k+1}}(D_r) \mid r \in \{0, \ldots, k\}\}$
  if $\tilde{P}_k = \tilde{M}_k = \tilde{M}_{k+1}$ **then**
    $\tilde{P}_{k+1} := \mathsf{null}$
  else
    $\tilde{P}_{k+1} :=$ lectically smallest subset of $\tilde{M}_{k+1}$ that is
    – lectically greater than $\tilde{P}_k$, and
    – respects all implications from $\tilde{\mathcal{L}}_{k+1} \cup \tilde{\mathcal{S}}_{k+1}$.
  end if
  $k := k + 1$
end while
return $\{\bigsqcap P_r \sqsubseteq D_r \mid r \in \{0, \ldots, k-1\}\}$

$i = (\Delta_i, \cdot^i)$. Let $U \subseteq M$ be a set of $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept descriptions and D a minimal possible consequence of $\bigsqcap U$ with respect to $\mathcal{O}$ for $\mathcal{N}_C$. If $\bigsqcap U \sqsubseteq D$ holds in i then D satisfies $\text{pr}_M(D) = U''$.

Proof. Corollary 4.13 yields $\text{pr}_M((\bigsqcap U)^{ii}) = U''$. Lemma 7.25 implies $(\bigsqcap U)^{ii} = D$. We obtain $\text{pr}_M(D) = U''$. □

We do not yet know if Algorithm 12 terminates. For now, we only assume that it terminates after n iterations of the outer while-loop. If it terminates the output of Algorithm 12 is the set of GCIs

$$\mathcal{B}_7 = \{\bigsqcap P_k \sqsubseteq D_k \mid k \in \{0, \dots, n-1\}\}.$$

Theorem 7.27. *If Algorithm 12 terminates after the n-th iteration of the outer while-loop, then its output $\mathcal{B}_7$ is a finite base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in the background model i.*

Proof. The idea of the proof is the same as in the completeness proof for Model Exploration (Theorem 6.16). We prove that Algorithm 12 with a representation $\mathcal{O}_0$ of i as input has the same output as Algorithm 10 with the full background model i as input. We use induction over k to show that $\tilde{P}_k \doteq \bar{P}_k$, $D_k \equiv (\bigsqcap \bar{P}_k)^{ii}$, $\tilde{M}_k \doteq \bar{M}_k$, $\tilde{\mathcal{L}}_k \doteq \bar{\mathcal{L}}_k$ and $\tilde{\mathcal{S}}_k \doteq \bar{\mathcal{S}}_k$ holds for all $k \in \{0, \dots, n\}$, where $\doteq$ denotes equality up to equivalence. The *base case* can be treated as in the proof of Theorem 6.16.

Step Case: We assume that $\tilde{P}_m \doteq \bar{P}_m$, $D_m \equiv (\bigsqcap \bar{P}_m)^{ii}$, $\tilde{M}_m \doteq \bar{M}_m$, $\tilde{\mathcal{L}}_m \doteq \bar{\mathcal{L}}_m$ and $\tilde{\mathcal{S}}_m \doteq \bar{\mathcal{S}}_m$ holds for all $m \leq k$. From $D_k \equiv (\bigsqcap \bar{P}_m)^{ii}$ we readily obtain $\tilde{M}_{k+1} \doteq \bar{M}_{k+1}$ and $\tilde{\mathcal{S}}_{k+1} \doteq \bar{\mathcal{S}}_{k+1}$. Lemma 7.26 and $\tilde{M}_k \doteq \bar{M}_k$ yield that

$$\text{pr}_{\tilde{M}_k}(D_r) \doteq \text{pr}_{\bar{M}_k}(D_r) = \bar{P}_r''^k$$

holds for all $r \leq k$. This proves $\tilde{\mathcal{L}}_k \doteq \bar{\mathcal{L}}_k$. The value of $\tilde{P}_k$ depends only on $\tilde{M}_k$, $\tilde{\mathcal{L}}_k$ and $\tilde{\mathcal{S}}_k$. It therefore holds that $\tilde{P}_k \doteq \bar{P}_k$. Algorithm 12 can only reach Line 11 if the expert has confirmed that $\bigsqcap \tilde{P}_k \sqsubseteq D_k$ holds in the background model i. Lemma 7.25 shows that $D_{k+1} \equiv (\bigsqcap \tilde{P}_{k+1})^{ii}$ holds and thus by induction hypothesis $D_{k+1} \equiv (\bigsqcap \bar{P}_{k+1})^{ii}$ holds.

We have thus shown that Algorithm 12 and Algorithm 10 produce the same output. Since the output of Algorithm 10 is correct, $\mathcal{B}_7$ must be correct as well. □

The proof of Theorem 7.27 in particular demonstrates that, outside the inner while-loop, Algorithm 12 exhibits the same behaviour as Algorithm 10. Hence, unless it remains in the inner while-loop, Algorithm 12 terminates.

Theorem 7.28. *The expert can ensure that Algorithm 12 terminates.*

Proof. Algorithm 12 terminates if it does not remain in the inner while-loop. Assume that the expert provides $\mathcal{O}_i = (\mathcal{T}_i, \mathcal{A}_i)$ from Section 7.1.2 as the new ontology in Line 7. This is possible since $\mathcal{O}_i$ a representation of i by Lemma 7.1 and contains an explicit counterexample to all GCIs that do not hold in i by Theorem 7.3. Then $D_k \in \mathfrak{mpc}_{\mathcal{O}_i}^{\mathcal{N}_C}(\bigsqcap \tilde{P}_k)$ is obtained as a possible consequence of $\bigsqcap \tilde{P}_k$ with respect to $\mathcal{O}_i$ for $\mathcal{N}_C$. In particular this means that there cannot be an explicit counterexample to the GCI $\bigsqcap \tilde{P}_k \sqsubseteq D_k$. Thus according to Theorem 7.3 the GCI $\bigsqcap \tilde{P}_k \sqsubseteq D_k$ holds in i, i. e. D_k satisfies $(\bigsqcap \tilde{P}_k)^i \subseteq D_k^i$. Lemma 4.1 yields $(\bigsqcap \tilde{P}_k)^{ii} \sqsubseteq D_k$. Since $(\bigsqcap \tilde{P}_k)^{ii}$ is also a possible consequence of $\bigsqcap \tilde{P}_k$ the minimality of D_k implies that $(\bigsqcap \tilde{P}_k)^{ii} \equiv D_k$ holds. Algorithm 12 will not remain in the inner while-loop and therefore terminates. □

Unfortunately, it is also possible for the expert to provide "bad" counterexamples, that allow the algorithm to remain in the inner while-loop. Example 7.5 illustrates this. This motivates us to restrict the ontology statements that can be added to those of type (7.9), (7.10), (7.11), (7.12), (7.13), and (7.14).

Corollary 7.29. *If the background model $i = (\Delta_i, \cdot^i)$ is finite and the expert is only allowed to add the elements of Δ_i as individual names and statements of type (7.9), (7.10), (7.11), (7.12), (7.13), and (7.14) to the ontology in Line 7 of Algorithm 12 then Algorithm 12 will terminate after finitely many iterations.*

Proof. Since there are only finitely many elements of Δ_i there are only finitely many statements of types (7.9) to (7.14). Therefore, after finitely many iterations the ontology $\mathcal{O}_l$ must contain all statements from $\mathcal{O}_i$ from Section 7.1.2. Then the arguments of Theorem 7.28 apply and prove that Algorithm 12 terminates. □

Allowing only the statements of types (7.9) to (7.14) may seem restrictive. However, it is not against the philosophy of exploration formalisms

to be restrictive with respect to the GCIs that can be added manually. The purpose of ABox Exploration is that the expert does not have to develop GCIs herself, but that GCIs are proposed by the system. The expert should not have to provide anything apart from "Yes/No"-answers and counterexamples. Therefore, *not allowing* the expert to provide more than this kind of information is not a severe restriction. All types of statements (7.9) to (7.14) have an intuitive meaning that can be reformulated in natural language. For example a GCI $\exists r.F_b \sqsubseteq F_a$ of type (7.14) can be reformulated to read that "a cannot have an r-successor that differs from b". Using natural language might make it even easier for the expert to provide counterexamples.

Example 7.5. This simple example illustrates that Algorithm 12 need not terminate if we allow the arbitrary ontological axioms to be added in Line 7. We consider the background model i over the signature $(\mathcal{N}_C, \mathcal{N}_R) = (\{A\}, \{r\})$ which is depicted in Figure 7.7. We assume that the ontology $\mathcal{O}_0$ consists of an empty TBox $\mathcal{T}_0 = \emptyset$ and an ABox $\mathcal{A}_0 = \{\top(x)\}$. The first left-hand side that Algorithm 12 comes up with is, as always, $\top$. The bottom concept $\bot$ is not a possible consequence of $\top$ since $\mathcal{O}_0$ cannot have an empty model. Instead, the minimal possible consequence of $\top$ is $C_{\text{all}} = (A_{\text{all}}, \mathcal{T}_{\text{all}})$ where $\mathcal{T}_{\text{all}}$ is the TBox

$$\mathcal{T}_{\text{all}} = \{A_{\text{all}} \equiv A \sqcap \exists r.A_{\text{all}}\}$$

(see Figure 7.8 for a witness model). In this special case the minimal possible consequence is even unique up to equivalence.
Thus the expert sees the question "Does the GCI

$$\top \sqsubseteq C_{\text{all}}$$

hold in i?". Since $x \in \Delta_i$ satisfies $x \notin A^i$ and thus also $x \notin {C_{\text{all}}}^i$ the expert rejects this GCI. She then tries to turn the individual x into a counterexample using the method that has been described in Section 7.1.1. Thus she adds a new concept name T_1 to the set of concept names, the GCI $T_1 \sqcap A \sqsubseteq \bot$ to the ABox, and the assertion $T_1(x)$ to the ABox. She has thereby stated that x is not an instance of A. Algorithm 12 then computes a new minimal possible consequence of $\top$, namely the description $\exists r.C_{\text{all}}$ (see Figure 7.9 for a witness model). The next GCI that is presented to the expert would thus be

$$\top \sqsubseteq \exists r.C_{\text{all}}.$$

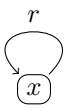

Figure 7.7: The Background Model i from Example 7.5

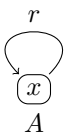

Figure 7.8: Witness Model

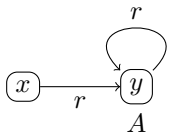

Figure 7.9: Witness Model

Once again, the expert refutes the GCI and turns x into a counterexample, by making sure that x is not an instance of $\exists r.A$. The next GCI is

$$\top \sqsubseteq \exists r.\exists r.C_{\text{all}}.$$

If the expert maintains her strategy, then this will continue indefinitely. This illustrates, that if we allow greater freedom to the expert, the ABox Exploration does not necessarily terminate.

Example 7.6. In the beginning of this chapter we have argued that in the background model from Figure 7.1 counterexamples in Model Exploration cannot be provided without feeding the entire background model into the algorithm. By contrast, we demonstrate how counterexamples can be described in ABox Exploration using only statements of the types (7.9) to (7.14). We present only the first iteration of the outer while-loop of Algorithm 12, not the entire exploration. The signature of the background model i is $(\{\mathsf{Male}, \mathsf{Female}\}, \{\mathsf{hasChild}, \mathsf{hasParent}, \mathsf{marriedTo}\})$. We assume that Algorithm 12 receives an empty ontology $\mathcal{O}_0 = (\emptyset, \emptyset)$ as input. The first GCI that the algorithm produces is

$$\top \sqsubseteq \bot.$$

This is refuted, and any individual can serve as a counterexample. For example, the expert might add Sally. To speed things up she could also describe some properties that Sally does not have, for example Sally is not male and does not have children. Using our restricted syntax this

can be achieved using the ontology $\mathcal{O}_1 = (\mathcal{T}_1, \mathcal{A}_1)$, where

$$\begin{aligned}\mathcal{T}_1 = \{&\mathsf{T}_{\mathsf{Sally}} \sqcap \mathsf{F}_{\mathsf{Sally}} \sqsubseteq \bot,\\ &\mathsf{Male} \sqsubseteq \mathsf{F}_{\mathsf{Sally}},\\ &\exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{F}_{\mathsf{Sally}}\}\end{aligned}$$

and

$$\mathcal{A}_1 = \{\mathsf{T}_{\mathsf{Sally}}(\mathsf{Sally})\}.$$

Thus, $\bot$ is not a possible consequence of $\top$ any more, nor is any concept description that is subsumed by Male or $\exists \mathsf{hasChild}.\top$. The new minimal possible consequence of $\top$ is

$$\mathsf{Female} \sqcap \exists \mathsf{marriedTo}.C_{\text{all}} \sqcap \exists \mathsf{hasParent}.C_{\text{all}},$$

where C_{all} is defined in analogy to (4.2). Obviously, the GCI

$$\top \sqsubseteq \mathsf{Female} \sqcap \exists \mathsf{marriedTo}.C_{\text{all}} \sqcap \exists \mathsf{hasParent}.C_{\text{all}},$$

still does not hold in the background model, and therefore the expert decides to add Don as a counterexample. She expresses that Don is not female, has no parents and is not married by adding the corresponding axioms to the ontology. This results in the minimal possible consequence of $\top$ being $\top$ which means that Algorithm 12 leaves the inner-while loop.

Notice that only two individuals had to be added so far. Moreover, it was not even necessary to describe all properties of these individuals. For example the ontology still does not contain the knowledge that Don is male, or that Sally has a female parent. By contrast, in Model Exploration the expert would have had to add the entire model already to refute the very first GCI.

7.3.2 Exploration Using Approximations

In the previous section we have seen that in order to guarantee termination of Algorithm 12 we can only allow statements of types (7.9), (7.10), (7.11), (7.12), (7.13), and (7.14) to be added to the ontology. We now look at the even simpler setting, where we also assume the original ontology $\mathcal{O}_0$ to be of this type. Apart from this assumption the input remains the same as for Algorithm 12, i. e. there is a background

model i over the signature $(\mathcal{N}_C, \mathcal{N}_R)$, the knowledge base $\mathcal{O}_0$ is a representation of i and the expert has complete knowledge about i. In such a setting minimal approximated possible consequences are unique and exhibit better computational behaviour than the actual minimal possible consequences (Corollary 7.23). We modify the ABox Exploration Algorithm 12 with replacing minimal possible consequences by minimal approximated consequences. The result is listed as Algorithm 13.

Notice, that it may happen that a given GCI $C \sqsubseteq D$ is refuted by the current ontology $\mathcal{O}_j$, but $\mathcal{O}_j$ does not contain an explicit counterexample to $C \sqsubseteq D$. In Algorithm 12 such a GCI would not be presented to the expert. In Algorithm 13 the GCI needs to be presented to the expert, who needs to provide an explicit counterexample. Thus using approximations may increase expert interaction.

Lemma 7.30. *Algorithm 13 terminates for all inputs $\mathcal{O}_0$ and for all background models i.*

Proof. We use a similar argument as is Lemma 6.15. Since we only allow statements of types (7.9) to (7.14) in the ontologies $\mathcal{O}_l$, and since there are only finitely many statements of these types after a certain number of iterations, $\mathcal{O}_l$ must contain the entire ontology $\mathcal{O}_i$ from Section 7.1.2. Theorem 7.3 yields that every GCI $C \sqsubseteq D$ that does not have an explicit counterexample in $\mathcal{O}_i$ must hold in i. Therefore if D is an approximated possible consequence of C the GCI $C \sqsubseteq D$ holds in i, i. e. $C^i \subseteq D^i$ holds. This yields $C^{ii} \sqsubseteq D$ by Lemma 4.1. Since C^{ii} is a possible consequence (i serves as a witness model), and therefore also an approximated possible consequence this proves that C^{ii} is the (unique) minimal approximated possible consequence of C with respect to $\mathcal{O}_i$. Consequentially, once $\mathcal{O}_l$ contains the entire ontology $\mathcal{O}_i$, Algorithm 13 must leave the inner while-loop. Outside the inner while-loop Algorithm 13 behaves exactly like Algorithm 12 and therefore terminates. □

The proof that when Algorithm 13 terminates after the n-th iteration its output

$$\{\bigsqcap P_r \sqsubseteq D_r \mid r \in \{0, \ldots, n\}\}$$

is a base for the GCIs holding in i can be done in analogy to the proof of Theorem 7.27. Again, one first proves in analogy to Lemma 7.25 that

Algorithm 13 ABox Exploration Using Approximated Possible Consequences

input $\mathcal{O}_0 := (\mathcal{T}_0, \mathcal{A}_0)$ {ontology $\mathcal{O}$}
$\tilde{P}_0 := \emptyset$, $\tilde{M}_0 := \mathcal{N}_P \cup \{\bot\}$, $k := 0$, $l := 0$,
$\tilde{\mathcal{S}}_0 := \{\{\bot\} \to \{A\} \mid A \in \mathcal{N}_P\}$, $\tilde{\mathcal{L}}_0 := \emptyset$
while $P_k \neq \mathsf{null}$ **do**
  $D_k := \mathfrak{mapc}^{\mathcal{N}_C}_{\mathcal{O}_l}(\bigsqcap \tilde{P}_k)$
  while $\mathcal{O}_l \cup \{\bigsqcap \tilde{P}_k \sqsubseteq D\}$ is inconsistent or expert refutes $\bigsqcap \tilde{P}_k \sqsubseteq D$ **do**
    Ask the expert for a new ontology $\mathcal{O}_{l+1}$ that
    – extends $\mathcal{O}_l$,
    – is a representation of the background model, and
    – contains an explicit counterexample for $\bigsqcap \tilde{P}_k \sqsubseteq D$
    $D_k := \mathfrak{mapc}^{\mathcal{N}_C}_{\mathcal{O}_{l+1}}(\bigsqcap \tilde{P}_k)$
    $l := l + 1$
  end while
  $\tilde{M}_{k+1} := \tilde{M}_k \cup \begin{cases} \{\exists r. \bigsqcap \tilde{P}_k \mid r \in \mathcal{N}_R\} & \text{if } D_\ell \not\equiv D_k \text{ for all } \ell < k \\ \emptyset & \text{otherwise} \end{cases}$
  $\tilde{\mathcal{S}}_{k+1} := \{\{A\} \to \{B\} \mid A, B \in \tilde{M}_{k+1}, A \sqsubseteq B\}$ {updating $\tilde{\mathcal{S}}_{k+1}$ and $\tilde{\mathcal{L}}_{k+1}$}
  $\tilde{\mathcal{L}}_{k+1} := \{\tilde{P}_r \to \mathrm{pr}_{\tilde{M}_{k+1}}(D_r) \mid r \in \{0, \ldots, k\}\}$
  if $\tilde{P}_k = \tilde{M}_k = \tilde{M}_{k+1}$ **then**
    $\tilde{P}_{k+1} := \mathsf{null}$
  else
    $\tilde{P}_{k+1} :=$ lectically smallest set subset of $\tilde{M}_{k+1}$ that is
    – lectically greater than $\tilde{P}_k$, and
    – respects all implications from $\tilde{\mathcal{L}}_{k+1} \cup \tilde{\mathcal{S}}_{k+1}$.
  end if
  $k := k + 1$
end while
return $\{\bigsqcap P_r \sqsubseteq D_r \mid r \in \{0, \ldots, k-1\}\}$

when the expert accepts a GCI we have found the correct right-hand side:

Lemma 7.31. *Let $\mathcal{O}$ be a representation of $i = (\Delta_i, \cdot^i)$. Let C be an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description and D a minimal approximated possible consequence of C with respect to $\mathcal{O}$ for $\mathcal{N}_C$. If $C \sqsubseteq D$ holds in i then D satisfies $D \equiv C^{ii}$.*

Proof. Using i as a witness model it follows that C^{ii} is a possible consequence of C. Thus it is also an approximated possible consequence of C and we obtain $D \sqsubseteq C^{ii}$. On the other hand, since $C \sqsubseteq D$ holds in i we obtain $C^i \subseteq D^i$. Lemma 4.1 yields $C^{ii} \sqsubseteq D$. □

Corollary 7.32. *If Algorithm 13 terminates after the n-th iteration of the outer while-loop, then its output*

$$\{\bigsqcap P_r \sqsubseteq D_r \mid r \in \{0, \ldots, n\}\}$$

is a finite base for the $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs holding in the background model i.

Proof. This can be shown in analogy to Theorem 7.27 by comparing the behaviour of Algorithm 13 to Algorithm 10. □

The advantages of Algorithm 13 over Algorithm 12 are that approximated minimal possible consequences can be computed more effectively than minimal possible consequences, and are moreover unique up to equivalence. A drawback is, that expert interaction may be necessary, even in some cases where it is already known that a certain GCI is refuted.

8 Related Work

This chapter presents several works that relate to this thesis. In the first section we introduce early attempts that bridge the gap between FCA and DL, or similar logics: Terminologic Attribute Logic and Logic Information Systems. Their ideas are related to what we call induced contexts and model-based most specific concepts in this work. The second section deals with the formalisms by Baader and Sertkaya, and by Rudolph. These have already been introduced in Section 1.3. Here, we look at them again from a low-level perspective and compare them to Model-Exploration and ABox-Exploration. Finally, two formalisms are presented that allow to combine general fixpoint semantics and general TBoxes. These are hybrid TBoxes, as introduced in [Bra06], and the logic $\mathcal{EL}^\nu$, as introduced in [LPW10a].

8.1 Bridging the Gap between FCA and Logics

We present two approaches, Terminological Attribute Logic and Logical Information Systems, that can be considered as first attempts to provide a common framework combining FCA and (Description) Logics. There has been other work at the intersection of FCA and DL, where FCA was mainly used as a tool for a specific problem in DL, for example to assist in the computation of the subsumption hierarchy of all conjunctions of a set of concepts [Baa95, BS04], the subsumption hierarchy of conjunctions and disjunctions [Stu96b] or the subsumption hierarchy of all least common subsumers [BM00]. Since these approaches are not as closely related to this work, we do not give a detailed presentation of them.

Table 8.1: Syntax and Semantics of Relation Terms

Name	Syntax	Semantics
relation names	$R \in \mathbf{R}$	$\bar{R}$
identity	id	$\{(g,g) \mid g \in G_{\mathbb{K}}\}$
inverse	S^d	$\{(h,g) \mid (g,h) \in [\![S]\!]_i\}$
complement	S^c	$G_{\mathbb{K}} \times G_{\mathbb{K}} \setminus [\![S]\!]_i$

Table 8.2: Syntax and Semantics of Attribute Terms

Name	Syntax	Semantics
attribute names	$m \in \mathbf{M}$	$\{g \in G_{\mathbb{K}} \mid (g, \bar{m}) \in I_{\mathbb{K}}\}$
top concept	$\top$	$G_{\mathbb{K}}$
bottom concept	$\bot$	$\emptyset$
negation	$\neg t$	$G_{\mathbb{K}} \setminus [\![t]\!]_i$
conjunction	$s \wedge t$	$[\![s]\!]_i \cap [\![t]\!]_i$
disjunction	$s \vee t$	$[\![s]\!]_i \cup [\![t]\!]_i$
existential restriction	$\exists R.t$	$\{g \in G_{\mathbb{K}} \mid \exists h \in [\![t]\!]_i \colon (g,h) \in [\![R]\!]_i\}$
value restriction	$\forall R.t$	$\{g \in G_{\mathbb{K}} \mid \forall h \in G_{\mathbb{K}} \colon (g,h) \in [\![R]\!]_i \Rightarrow h \in [\![t]\!]_i\}$

8.1.1 Logical Scaling and Terminological Attribute Logic

Serious attempts at combining FCA and logic started in the late 1990s with the works of Susanne Prediger [Pre97, PS99, Pre00]. Prediger's primary interest was *logical scaling*, a method to transform many-valued formal contexts (concepts where an object/attribute pair can be assigned more than two values) into normal (also called two-valued) formal contexts. Since many-valued contexts are not relevant to this work, we only give an intuitive description of the results that have been obtained in this area. In [Pre00] Prediger presents a logic that is tailored to be

used with formal contexts. She presents two variants of this logic, one for two-valued contexts and one for many-valued logics. The logic for two-valued contexts can be viewed as a syntactic variant of $\mathcal{ALCI}$ with negation of roles and identity. It is introduced as a tool to extend a context by defining new attributes, which yields an induced context. The intended purpose of the variant for many-valued contexts is logical scaling. Earlier works have used standard $\mathcal{ALC}$ and SQL as the language that is used to describe logical scales [Pre97, PS99]. A similar approach with a logic that does not use quantifiers exists under the name *Contextual Attribute Logic* [GW99] and an approach using full first order logic has been presented in [Zic91].

Two-Valued Contexts

Terminological attribute logic for two-valued contexts, as Prediger calls her logic, can be viewed as a syntactic variant of the description logic $\mathcal{ALCI}^{\neg,id}$. The most obvious difference is that it uses *relational contexts* to encode the information that is normally provided by an interpretation.

Definition 8.1 (Relational Context)**.** A tuple $((G_\mathbb{K}, \mathcal{R}_\mathbb{K}), M_\mathbb{K}, I_\mathbb{K})$ where $G_\mathbb{K}$ and $M_\mathbb{K}$ are finite sets, $\mathcal{R}_\mathbb{K}$ is a finite set of relations $R_\mathbb{K} \subseteq G_\mathbb{K} \times G_\mathbb{K}$, and $I_\mathbb{K}$ is a relation $I_\mathbb{K} \subseteq G_\mathbb{K} \times M_\mathbb{K}$ is called *relational context*.

Terminological Attribute Logic starts with an alphabet of attribute names $\mathbf{M}$ and an alphabet of relation names $\mathbf{R}$.[1] From the two alphabets one can create new *attribute terms* and *relation terms* inductively, using constructors from Table 8.1 and Table 8.2. The semantics is defined using a relational context $((G_\mathbb{K}, \mathcal{R}_\mathbb{K}), M_\mathbb{K}, I_\mathbb{K})$ and a mapping $\bar{\cdot} : \mathbf{M} \cup \mathbf{R} \to M_\mathbb{K} \cup \mathcal{R}_\mathbb{K}$. The relational context together with $\bar{\cdot}$ gives rise to a mapping $[\![\cdot]\!]_i$ that maps role terms to binary relations on G and attribute terms to subsets of G. $[\![\cdot]\!]_i$ is defined according to Table 8.1 and Table 8.2. The correspondence to $\mathcal{ALCI}^{\neg,id}$ is evident.

One of the applications of Terminological Attribute Logic for two-valued contexts are induced contexts. It allows new attributes to be

[1] There is a natural correspondence between attribute names and relation names in Terminological Attribute Logic, and concept names and role names in DL, respectively. It should be noticed, however, that attribute names and attribute terms are by convention denoted by lower case letters while relation names and relation terms are denoted by upper case letters.

Table 8.3: A Formal Context about Cheeses

	softCheese	pricePerKG
Roquefort	×	19
Morbier		12
Gruyère		10
Emmental		8
Brie	×	7
Camembert	×	8

Table 8.4: The Logically Scaled Context about Cheeses

	softCheese	cheap
Roquefort	×	
Morbier		
Gruyère		
Emmental		×
Brie	×	×
Camembert	×	×

defined using a formal semantics. Apart from the obvious differences in syntax and expressivity they are the same as the induced contexts presented in Section 4.2 for $\mathcal{EL}^{\bot}_{\text{gfp}}$.

Logical Scaling

A *many-valued context* is a 4-tuple $\mathbb{K} = (G, M, W, I)$ where G is a set of objects, M is a set of attributes, W is a set of values, and I is a mapping $I : G \times M \to W$ that assigns a value to each pair $(g, m) \in G \times M$ (eg. Table 8.3). A substantial amount of FCA research has been dedicated to two-valued contexts. In order to benefit from this research transformations from many-valued contexts into two-valued contexts have emerged. For example, we might transform the context about cheeses from Ta-

ble 8.3 to a two-valued context by associating a certain price range with an attribute cheap. The usual approach is to provide a *conceptual scale*. A conceptual scale is a formal context $\mathbb{K}_S = (G_S, M_S, I_S)$ that uses a fresh set of attributes $M_S \cap M = \emptyset$ and whose objects are values of the original context $G_S \subseteq W$. To transform the many-valued context $\mathbb{K} = (G, M, W, I)$ into a two-valued context a scale is associated with each attribute $m \in M$. In the resulting *scaled context* an object $g \in G$ has the attribute $m_S \in M_S$ if and only if $(g, m)Iw$ holds and w has the attribute m_S in the scale $\mathbb{K}_S$ that corresponds to m. Of course, the transformation to a scaled context is hardly ever lossless.

Providing a formal context as a scale for each attribute can be impractical or even impossible, when the set of values is large or infinite, as is often the case with numerical values. Therefore, alternative methods for providing scales have been examined. Prediger's contributions include the method of *logical scaling*.

The method of logical scaling uses Terminological Attribute Logic for many-valued contexts. This logic can be used to build attribute descriptions which are statements such as "A cheap cheese is a cheese that is either a softCheese and has a pricePerKG of less than 10 or that is not a softCheese and has a pricePerKG of less than 15." Attribute descriptions can then be used to define attributes of a two-valued context, very much in the sense of induced contexts (c. f. Table 8.4). We are not going to introduce Terminological Attribute Logic for many-valued contexts formally. We would only like to point out two interesting facts. First, its semantics do not use relations on the set of objects, instead relations on the set of values are used. Second, it is entirely quantifier free. The latter results mainly from the fact that each object-attribute pair is assigned exactly one value, which is why there is no need for quantification.

Power Context Families

In [PW99] *power context families*, which are a generalization of relational contexts, are presented. A power context family is a family of formal contexts

$$\{\mathbb{K}_k = (G_k, M_k, I_k)\}_{k \in \{0,\ldots,n\}}$$

where $G_k \subseteq G_0^k$ holds for all $k \in \{0, \ldots, n\}$. Thus, power contexts allow not only for binary, but for n-ary relations where n can be arbitrarily

large. A relational context can be viewed as a special case of a power context family where $n = 2$. In the field of power context families Prediger delivers two important contributions. First, she lifts basic FCA notions such as concept lattices to the level of power context families. Second, she devises a method for obtaining a power context family from a many-valued context which is called relational scaling.

8.1.2 Logic Information Systems

The objective of Logic Information Systems is to provide a method for users to query and navigate through data by using a combination of DL and FCA [Fer02, FR04]. They are based on a theory called Logical Concept Analysis [FR00]. Logical Concept Analysis is related to this thesis as it introduces an operator that behaves similar to model-based most specific concepts.

Logical Concept Analysis

The idea of logical concept analysis is to avoid FCA attributes entirely, and to replace them by a single logical description for each object. Logical Concept Analysis requires a DL $\mathcal{L}$ that must at least allow for conjunction $\sqcap$, disjunction $\sqcup$, the top concept $\top$ and the bottom concept $\bot$. We denote the set of all $\mathcal{L}$-concept descriptions by $\mathcal{L}$, as well. The set of concept descriptions $\mathcal{L}$ together with subsumption $\sqsubseteq$ forms a bounded lattice $(\mathcal{L}, \sqsubseteq)$ where $\sqcap$ is the meet operator, $\sqcup$ is the join operator, and $\top$ and $\bot$ are the greatest and least element, respectively. It is remarkable that the semantics of $\mathcal{L}$ are used only to obtain the lattice structure and not in any of the following technical definitions. The only reason why a logic and not an arbitrary bounded lattice is used is that logics allow for more intuitive descriptions in Logical Information Systems. Ferré describes the logic as being the equivalent to a schema in databases.

As mentioned before, attributes are replaced by a single logical description for each object. Thus a *logical context* is a triple $\mathbb{K} = (G, \mathcal{L}, d)$ where G is a finite set of objects and d is a function $d : G \to \mathcal{L}$ that maps each object to a concept description that describes it. Furthermore, the

operators $\sigma_{\mathbb{K}}$ and $\tau_{\mathbb{K}}$ are defined as

$$\begin{aligned}\sigma_{\mathbb{K}} &: 2^G \to \mathcal{L} \\ &\quad A \mapsto \bigsqcup_{g \in A} d(g) \\ \tau_{\mathbb{K}} &: \mathcal{L} \to 2^G \\ &\quad C \mapsto \{g \in G \mid d(g) \sqsubseteq C\}.\end{aligned}$$

The intuition behind $\tau_{\mathbb{K}}$ is that it maps every concept description C to the set of objects that belong to the concept C. Clearly, every object g is an element of $\tau_{\mathbb{K}}(d(g))$ and there cannot be a concept description C to which g belongs that is more specific than $d(g)$. Thus the concept description $d(g)$ has the flavor of a most specific concept. In logics that provide for disjunction the least common subsumer of two concept descriptions is their disjunction. We can think of $\sigma_{\mathbb{K}}$ as a function that maps a set $A \subseteq G$ to the most specific concept description C to which all elements of A belong, i. e. which satisfies $A \subseteq \tau_{\mathbb{K}}(C)$. This exposes the common ideas behind logical contexts and model-based most specific concepts. The most obvious difference between the two is that model-based most specific concepts are obtained from a model while the mapping d of a logical context must be given explicitly.

The basic notions of FCA can be replicated in the setting of logical contexts. Ferré defines

- *logical concepts*, which are pairs $(A, C) \in 2^G \times \mathcal{L}$ where $A = \tau_{\mathbb{K}}(C)$ and $C = \sigma_{\mathbb{K}}(A)$, and
- *contextualized subsumption*, where C is said to contextually subsume D if $\tau_{\mathbb{K}}(C) \subseteq \tau_{\mathbb{K}}(D)$ holds.

Furthermore, Ferré defines *feature contexts* which closely ressemble induced contexts: If $\mathbb{K} = (G, \mathcal{L}, d)$ is a logical context and $\mathcal{F} \subseteq \mathcal{L}$ is a set of concept descriptions, then the formal context $\mathbb{K}_{\mathcal{F}} = (G, \mathcal{F}, I_{\mathcal{F}})$, where $gI_{\mathcal{F}}C$ if and only if $g \in \tau_{\mathbb{K}}(C)$, is called the *feature context* of $\mathcal{F}$ and $\mathbb{K}$.

Logical Information Systems

Logical Contexts have been designed with Logical Information Systems as their application in mind. Logical Information Systems are intended

to facilitate querying and navigation in data. In principle a Logical Information System is a logical context, together with some reasoning services of which the two most important ones are querying and navigation.

Querying is defined in a relatively straightforward way: Given a concept description $C \in \mathcal{L}$ one looks for all objects that belong to C. It consists simply of computing $\tau_{\mathbb{K}}(C)$. Navigation on the other hand is less obvious. The idea is that instead of writing a complex query manually the user may want to navigate through the data, gradually refining the search query. To narrow the search space, a set of features $\mathcal{F}$ is given, from which the queries are obtained. In principle, refinement of a query C means moving down one node in the concept lattice of the feature context $\mathbb{K}_{\mathcal{F}}$. What the navigation reasoning service does is to suggest so-called *links*. A link is a feature D, such that the conjunction of the current query C and D yields a concept description $C \sqcap D$ that satisfies $\tau_{\mathbb{K}}(C \sqcap D) \subsetneq \tau_{\mathbb{K}}(C)$. Using links the user can gradually move down the edges of the lattice until she arrives at the desired query.

8.2 Exploration Formalisms in DL

8.2.1 FCA-based Ontology Completion and OntoComp

In [BGSS07, Ser07] an ontology completion formalism is introduced. Like in this thesis, it is assumed that the expert has complete knowledge about the domain. Like Abox-Exploration the input of the algorithm is an ontology $\mathcal{O}_0 = (\mathcal{T}_0, \mathcal{A}_0)$. The difference between the two approaches is that [BGSS07] uses a weaker notion of completeness. In this formalism the expert is initially asked to select a set M of concept names that she deems interesting. The formalism's objective is to find a set of GCIs $\mathcal{B}$ such that

- all GCIs from $\mathcal{B}$ hold in the domain, and
- all GCIs of the form
$$\bigsqcap L \sqsubseteq \bigsqcap R,$$
where $L, R \subseteq M$, that hold in the domain follow from $\mathcal{B}$.

Hence, the main difference between this formalism and ours is that while we strive for completeness with respect to all $\mathcal{EL}^{\bot}$-GCIs Baader et al. only strive for completeness with respect to GCIs of a restricted form.

The main achievement of [BGSS07] is that they find a practical way of dealing with the open-world knowledge from the ontology in an FCA context. To this purpose they introduce the notion of a *partial context*.

Definition 8.2 (Partial Context)**.** A *partial object description* is a pair (A, S), where $A, S \subseteq M$ are disjoint subsets of M. A *partial context* is a set of partial object descriptions.

As the name says, a partial object description partially describes an object. Intuitively, if (A, S) is a partial object description then A contains the attributes that the object is known to have, while S contains the attributes that the object is known not to have. For all other attributes in $M \setminus (A \cup S)$ it is not known whether the object has them or not. Partial contexts can be visualized as cross tables, where each row represents one partial object description and the columns represent the attributes. In each cell of the table, there is a cross, an empty space, or a question mark depending on whether the object has the attribute, does not have the attribute, or whether it is unknown.

The ontology $\mathcal{O}$ together with the set M of concept names gives rise to the following partial context.

- Every individual $a \in \mathcal{N}_I$ yields the partial object description (A_a, S_a) where $A_a = \{C \in M \mid \mathcal{O} \models C(a)\}$ and $S_a = \{C \in M \mid \mathcal{O} \models \neg C(a)\}$.
- The whole ABox induces the partial context

$$\mathbb{K}_p = \{(A_a, S_a) \mid a \in \mathcal{N}_I\}. \tag{8.1}$$

This construction is very similar to the induced contexts from Section 4.2. The only difference is that partial contexts instead of conventional formal contexts are required to take care of the open-world semantics of the ontology $\mathcal{O}$.

In these partial contexts it is not immediately clear how to compute the right-hand sides of implications, since the FCA derivation operators do not exist in partial contexts. The following construction is proposed

Algorithm 14 Ontology Completion According to Baader et al.

input $\mathcal{O}$, M
$\mathbb{K}_0 :=$ partial context obtained from M and $\mathcal{O}$ according to (8.1)
$\mathcal{L}_0 := \emptyset$, $P_0 := \emptyset$, $k := 0$, $l := 0$ {Initialization}
while $P_k \neq M$ **do**
  while expert refutes $P_k \to \mathbb{K}_l(P_k)$ **do**
    ask the expert for a new partial context $\mathbb{K}_{l+1}$ that
    – extends $\mathbb{K}_l$,
    – contains a counterexample for $P_k \to \mathbb{K}_l(P_k)$
    $l := l + 1$
  end while
  $\mathcal{L}_{k+1} := \mathcal{L}_k \cup \{P_j \to \mathbb{K}_l(P_j)\}$
  $P_{k+1} :=$ lectically smallest subset of M that is
  – lectically greater than P_k, and
  – respects all implications from $\mathcal{L}_k$.
  $k := k + 1$
end while
return $\mathcal{L}_k$

as a replacement. Let $\mathbb{K}_p$ be a partial context and $P \subseteq M$ a set of attributes. Then $\mathbb{K}_p(P)$ is defined to be the set

$$\mathbb{K}_p(P) = M \setminus \bigcup \{S \mid (A, S) \in \mathbb{K}_p, P \subseteq A\}.$$

It can be shown that $\mathbb{K}_p(P)$ is the largest subset of M such that $\mathbb{K}_P$ does not contain a counterexample to $P \to \mathbb{K}_p(P)$. We would like to point out that in the special case where

- $\mathbb{K}_p$ is obtained from an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ and M according to (8.1),
- there are no dependencies among the attributes in M, e. g. because $\mathcal{T}$ is empty,

$\bigsqcap \mathbb{K}_p(P)$ is a minimal possible consequence of $\bigsqcap P$ with respect to $\mathcal{O}$ computed in the simple DL that allows only for conjunction and no other constructors. If there are dependencies among the attributes in

M then $\bigsqcap \mathbb{K}_p(P)$ corresponds to the approximated minimal possible consequence. Algorithm 14 shows the entire ontology completion algorithm, slightly rearranged to emphasize parallels to our algorithms.

8.2.2 Relational Exploration

Certainly, the theory that is most closely related to this work is the exploration formalism by Rudolph [Rud04, Rud06, RVH07]. Like in this work, Rudolph develops a method to obtain a set of GCIs from a possibly incomplete model. Eventually, the set of GCIs obtained is shown to be correct and complete (in a certain sense) for the GCIs holding in what Rudolph calls the "universe".

As the datastructure in which the counterexamples are provided, Rudolph uses the power context families of order 2, which we have introduced in Section 8.1.1, the section about Prediger's work. We have also mentioned that power context families of order 2 contain the same information as DL models. In order to not confuse the reader with another notation, we present Relational Exploration with DL models as the underlying datastructure and use the standard notation for concept descriptions.

In most of his works Rudolph uses the logic $\mathcal{FLE}$ which is $\mathcal{EL}^\bot$ extended with value restrictions $\forall$. Here, we present the method for $\mathcal{FLE}$ based on [Rud04]. In contrast to Model-Exploration Relational Exploration employs a method where role depth is increased step by step.

In order to axiomatize a given model i Relational Exploration starts with the context $\mathbb{K}_0$ that is induced by $\{\bot\} \cup \mathcal{N}_C$ and i. The Duquenne-Guigues-Base $\mathcal{B}_0$ as well as the set of all concept intents $\mathcal{H}_0$ of $\mathbb{K}_0$ is computed in the usual way. No attributes are added during the computation. Once this has been completed a new context is created.

Assume that $\mathcal{B}_k$ and $\mathcal{H}_k$ have been computed for the context $\mathbb{K}_k$. The new set of attributes M_{k+1} is defined as

$$\begin{aligned} M_{k+1} = & \mathcal{N}_C \\ & \cup \{\exists r. \bigsqcap U \mid U \in \mathcal{H}_k, r \in \mathcal{N}_R\} \\ & \cup \{\forall r. C \mid C \in M_k, r \in \mathcal{N}_R\}. \end{aligned} \tag{8.2}$$

The new context $\mathbb{K}_{k+1}$ is the context induced by M_{k+1} and i. Now, Next-Closure can be applied again, in order to obtain $\mathcal{B}_{k+1}$ and $\mathcal{H}_{k+1}$.

The algorithm terminates when $|\mathcal{H}_k| = |\mathcal{H}_{k+1}|$ holds. Rudolph has shown that this must be the case after finitely many iterations for every input model i.

Algorithm 15 Axiomatizing a Finite Model According to Rudolph

$k := 0$; $M_0 := \{\bot\} \cup \mathcal{N}_C$;
while $|\mathcal{H}_{k-1}| > |\mathcal{H}_{k-2}|$ **do**
  $K_k :=$ context induced by M_k and i
  Compute Duquenne-Guigues Base $\mathcal{B}_k$ and set of concept intents $\mathcal{H}_k$ of $\mathbb{K}_k$
  Obtain new set of attributes M_{k+1} from (8.2)
  $k := k + 1$
end while
return $\{\mathcal{B}_k\}_{k\in\{0,\dots,n\}}$

Upon termination the algorithm returns the family $\{\mathcal{B}_k\}_{k\in\{0,\dots,n\}}$. Rudolph shows that this is complete in the sense that it is possible to decide for any $\mathcal{FLE}$-GCI $C \sqsubseteq D$ whether $C \sqsubseteq D$ holds in i using only $\{\mathcal{B}_k\}_{k\in\{0,\dots,n\}}$ and no additional information. The decision procedure is not straightforward and cannot be performed by a DL reasoner. It uses a family of mappings $\{\varphi_k\}_{k\in\{0,\dots,n\}}$ that is defined inductively as follows. Each mapping φ_k maps arbitrary $\mathcal{FLE}$-concept descriptions of role depth less than or equal to k to subsets of M_k.

$$\varphi_0(A) = \{A\}, \text{ for all } A \in \{\bot\} \cup \mathcal{N}_C$$

and

$$\begin{aligned}
\varphi_{k+1}(A) &= \{A\}, \text{ for all } A \in \{\bot\} \cup \mathcal{N}_C \\
\varphi_{k+1}(\exists r.C) &= \{\exists r.\textstyle\bigsqcap(\varphi_k(C))''_k\} \\
\varphi_{k+1}(\forall r.C) &= \{\forall r.D \mid D \in \varphi_k(C)\} \\
\varphi_{k+1}(\textstyle\bigsqcap \mathcal{C}) &= \bigcup\{\varphi_k(C) \mid C \in \mathcal{C}\}.
\end{aligned}$$

Since the closure operator $\cdot''_k$ can be obtained as the implicational closure with respect to $\mathcal{B}_k$ the function φ_k can be computed without using any information except $\{\mathcal{B}_k\}_{k\in\{0,\dots,n\}}$. Rudolph shows that if C and D

are $\mathcal{FLE}$-concept descriptions of role depth at most n then $C \sqsubseteq D$ holds in i if and only if $(\varphi_n(D))''_n \subseteq (\varphi_n(C))''_n$. If C and D have a role depth that is larger than n then the decision procedure becomes more difficult. Rudolph presents another function π that is also defined inductively and whose exact definition we omit here. Like φ_k the function π can be computed using only $\{\mathcal{B}_k\}_{k\in\{0,\dots,n\}}$ and no additional information from i. It maps every $\mathcal{FLE}$-concept description C to a concept description $\pi(C)$ such that

- $\pi(C)$ has at most role depth n, and
- $\pi(C)^i = C^i$ holds.

To verify whether some GCI $C \sqsubseteq D$ holds in i it suffices to check whether $\pi(C) \sqsubseteq \pi(D)$ holds in i. This can be decided like for concept descriptions of role depth less than n.

In the fifth chapter of his dissertation Rudolph gives a high-level description of an exploration formalism that is based on the above procedure [Rud06]. The algorithm is in principle Algorithm 8.2.2 where instead of computing the Duquenne-Guigues Base of $\mathbb{K}_k$ in Line 4 a normal FCA Attribute-Exploration is performed on $\mathbb{K}_k$.

There are several issues that can be criticized in Rudolph's approach. It does not produce a base of the GCIs holding in i in the sense of Definition 5.1. The output is a family of implications and a rather complicated decision procedure. It is not clear whether a base can be obtained easily from the family $\{\mathcal{B}_k\}_{k\in\{1,\dots,n\}}$, and therefore it cannot be added immediately to a knowledge base. Secondly, the number of attributes in $\mathbb{K}_k$ is of order $\mathcal{O}(|\mathcal{N}_R|^k)$ and therefore grows exponentially. However, this is in part due to attributes that are of the form $\forall r.X$ and comparing the number of new attributes in Relational Exploration to our approach would not be fair. There is, however, a computational disadvantage not only with respect to the number of new attributes, but also with respect to the amount of expert interaction. By increasing role depth successively, one loses the property that only one GCI per left-hand side needs to be added. If the right-hand side is a cyclic model-based most specific concept in Model-Exploration the same left-hand side occurs in a question in each iteration of Relational Exploration until termination. This is likely to result in an increase in expert interaction. A last criticism is that Rudolph does not go into details of the exploration

formalism. Especially, the question in what form counterexamples have to be provided is not addressed. Since it is implicitly assumed that the induced contexts do not contain counterexamples to GCIs that hold in the "universe" and the induced contexts have a closed-world semantics, it stands to reason that the same problems regarding connectedness of the submodels can occur as in Model-Exploration. However, no details are found in Rudolph's work.

Advantages of Relational Exploration over Model-Exploration are that the questions that are asked do not contain cyclic concept descriptions and that it additionally allows for value restrictions.

8.3 $\mathcal{EL}$ and Fixpoint Semantics

8.3.1 $\mathcal{EL}$ with Hybrid TBoxes

$\mathcal{EL}$ with hybrid TBoxes is a logic that combines descriptive semantics and greatest-fixpoint semantics [BM05, Bra06]. It is motivated by the need for greatest-fixpoint semantics for non-standard reasoning services such as the least common subsumer or the most specific concept. At the same time one does not want to lose the usability of GCIs with descriptive semantics. We introduce $\mathcal{EL}$ with hybrid TBoxes. Unfortunately, while the combination of descriptive semantics and greatest-fixpoint semantics also occurs in $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCIs the results from $\mathcal{EL}$ with hybrid TBoxes cannot be used.

Definition 8.3 (Hybrid TBox)**.** Let $\mathcal{N}_P$ and $\mathcal{N}_D$ be sets of concept names and let $\mathcal{N}_R$ be a set of role names. The pair $(\mathcal{F}, \mathcal{T})$ is called a *hybrid $\mathcal{EL}$-TBox* if

- $\mathcal{F}$ is a general TBox using $\mathcal{N}_P$ as the set of concept names and $\mathcal{N}_R$ as the set of role names, and
- $\mathcal{T}$ is a cyclic TBox using $\mathcal{N}_P$ as the set of primitive concept names, $\mathcal{N}_D$ as the set of defined concept names, and $\mathcal{N}_R$ as the set of role names.

Definition 8.4 (Semantics of Hybrid TBoxes)**.** An interpretation $i = (\Delta_i, \cdot^i)$ is called a model of $(\mathcal{F}, \mathcal{T})$ if there is an interpretation $j = (\Delta_j, \cdot^j)$ such that

- j is a model of $\mathcal{F}$, and
- i is the gfp-model of $\mathcal{T}$ that corresponds to j, when j is viewed as a primitive interpretation of $\mathcal{T}$.

Therefore, the idea behind hybrid TBoxes is that the general TBox $\mathcal{F}$ is used to define simple concepts and then more complex concepts are built on top of these using gfp-semantics. That is why $\mathcal{F}$ is called the *foundation* of $(\mathcal{F}, \mathcal{T})$. Brandt's main result is that like in the non-hybrid versions of $\mathcal{EL}$ subsumption reasoning is tractable [BM05]. A polynomial time reasoning procedure has been implemented in an experimental reasoner called *Hyp* [BNS08]. Furthermore, Brandt provides algorithms for non-standard reasoning tasks such as least common subsumers [Bra06].

In hybrid TBoxes the defined concepts cannot be used in the descriptive part, i. e. in the foundation $\mathcal{F}$. This contrasts to an $\mathcal{EL}^{\bot}_{\text{gfp}}$-GCI $C \sqsubseteq D$ where $C = (A_C, \mathcal{T}_C)$ and $D = (A_D, \mathcal{T}_D)$. The GCI $C \sqsubseteq D$ is essentially a statement about the defined concept names A_C and A_D (because the interpretation of C and D is defined to be the interpretation of A_C and A_D in the corresponding gfp-models). Therefore, in order to be useful for reasoning in general $\mathcal{EL}^{\bot}_{\text{gfp}}$-TBoxes hybrid TBox would have to allow defined concept names in the foundation $\mathcal{F}$. Such a logic has not been examined yet.

8.3.2 $\mathcal{EL}^{\nu}$ and $\mathcal{EL}^{\nu+}$

The description logics $\mathcal{EL}^{\nu}$ and $\mathcal{EL}^{\nu+}$ make use of the close ties between modal logics and DL [Sch91]. They have been introduced in [LPW10b, LPW10a]. The idea behind $\mathcal{EL}^{\nu}$ is to use the greatest fixpoint operator ν from modal μ-calculus [BS07] explicitly in the concept descriptions. Thus the syntax of $\mathcal{EL}^{\nu}$ is the same as the syntax of classical $\mathcal{EL}$ but additionally allows the greatest fixpoint construction $\nu X.C$, where C is an $\mathcal{EL}^{\nu}$ concept description and X is a *free concept variable*. Concept variables are names from a set $\mathcal{N}_V$ that can be used within concept descriptions just like concept names. A variable is *free* if it occurs outside the scope of a ν-operator. An $\mathcal{EL}^{\nu}$ concept is *closed* if it does not contain any free variables. The semantics of the fixpoint operator is as follows. Let i be a interpretation and let v be an assignment that maps variables to subsets of Δ_i. By $v[X \to W]$ denote v modified by setting $v(X) = W$.

Then

$$(\nu X.C)^{i,v} = \bigcup\{W \subseteq \Delta_i \mid W \subseteq C^{i,v[X\to W]}\}.$$

While $\mathcal{EL}^\nu$ does provide a fixpoint constructor it still differs from $\mathcal{EL}_{\text{gfp}}$ in one aspect. The ν-operators in $\mathcal{EL}^\nu$ are evaluated successively, while the fixpoints of an $\mathcal{EL}_{\text{gfp}}$-TBox are evaluated simultaneously for all defined concept names. Lutz et al. consider a second logic $\mathcal{EL}^{\nu+}$ which allows for simultaneous fixpoint operators. Syntactically, $\mathcal{EL}^{\nu+}$ is $\mathcal{EL}$ extended by a constructor of the form

$$\nu_k X_1 \cdots X_n.C_1, \ldots, C_n$$

where $1 \leq k \leq n$. Its semantics is defined by

$$\begin{aligned}(\nu_k X_1 \cdots X_n.C_1, \ldots, C_n)^{i,v} &= \\ = \bigcup\{W_k \mid \exists W_1, \ldots, W_{k-1}, W_{k+1}, \ldots, W_n\colon \forall 1 \leq j \leq n\colon & \\ W_j \subseteq C_j^{i,v[X_1\to W_1,\ldots,X_n\to W_n]}\}.&\end{aligned}$$

Lutz et al. compare the expressivity of $\mathcal{EL}^{\nu+}$ and $\mathcal{EL}_{\text{gfp}}$.

Lemma 8.1. *$\mathcal{EL}^{\nu+}$ is strictly more expressive than $\mathcal{EL}_{\text{gfp}}$. More precisely, for every $\mathcal{EL}_{\text{gfp}}$-concept description there is an equivalent $\mathcal{EL}^{\nu+}$-concept description of polynomial size, but not every $\mathcal{EL}^{\nu+}$-concept description is equivalent to an $\mathcal{EL}_{\text{gfp}}$-concept description.*

Furthermore, a result is obtained stating that $\mathcal{EL}^\nu$ and $\mathcal{EL}^{\nu+}$ have the same expressivity, but $\mathcal{EL}^{\nu+}$ is exponentially more succinct. Further results include the proof of the existence of least common subsumers and most specific concepts in $\mathcal{EL}^{\nu+}$. The ideas for these proofs are adaptations of the techniques used for $\mathcal{EL}_{\text{gfp}}$. Most importantly they prove that standard reasoning problems such as consistency checking, subsumption with respect to TBoxes and the instance problem remain tractable within $\mathcal{EL}^{\nu+}$. Together with Lemma 8.1 this yields, that standard reasoning is also tractable in $\mathcal{EL}_{\text{gfp}}$.

These results and others obtained by Lutz et al. make a strong case for using $\mathcal{EL}^{\nu+}$ as the standard when an extension of $\mathcal{EL}$ with fixpoints is required. It is a possible direction of future work to examine whether the results from this thesis can be adapted to $\mathcal{EL}^{\nu+}$. We conjecture that this is not very difficult.

9 Conclusions

Model-Based Most Specific Concepts Almost all previous attempts at combining FCA and DL have used a construction that is similar to induced contexts. Therefore they have essentially used some variant of classical FCA in a context whose attributes are DL concept descriptions. We have provided an alternative approach, where we have introduced DL reasoning services that have similar properties as the derivation operators from FCA. These reasoning services are on the one hand the interpretation function of a model as a replacement for the derivation operator that maps sets of attributes to sets of objects, and on the other hand the model-based most specific concept as a replacement for the derivation operator that maps sets of objects to sets of attributes. They provide a framework that allows us to transfer several basic results from FCA to the DL world. One such result is that when searching for a base for the GCIs holding in a given model it suffices to consider right-hand sides that can be written as model-based most specific concepts.

We have dealt with the question whether model-based most specific concepts always exist. Unfortunately, this is not true for $\mathcal{EL}^{\bot}$. However, by extending $\mathcal{EL}^{\bot}$ by cyclic concept description and greatest-fixpoint semantics one can guarantee existence of model-based most specific concepts. They can be computed effectively: For a singleton set the model-based most specific concept is obtained via a simple linear time translation from the model to an $\mathcal{EL}^{\bot}_{\text{gfp}}$-concept description. For larger sets it is the least common subsumer of the model-based most specific concepts of its singleton subsets.

Finding a Finite Base Compared to earlier formalisms we adopt a stronger notion of completeness. We say that a set $\mathcal{B}$ of GCIs is complete for a model i if all GCIs that hold in i follow from the GCIs in $\mathcal{B}$. In the previous work [BGSS07] restrictions on the structure of the GCIs were made, we make no such restrictions.

We have addressed the question whether a finite base for the GCIs holding in a finite model can exist. Both for the logics $\mathcal{EL}^{\perp}_{\text{gfp}}$ and $\mathcal{EL}^{\perp}$ a construction for a finite base has been presented. The first step in this construction is the computation of a set of interesting concept descriptions M_i. This can be achieved using a version of Next-Closure. In a next step the context induced by M_i and i is constructed. We have shown that every implicational base of this induced context gives rise to a base for the $\mathcal{EL}^{\perp}_{\text{gfp}}$-GCIs holding in i, and the Duquenne-Guigues base of this induced context gives rise to a base with minimal cardinality. This base can then be transformed into an $\mathcal{EL}^{\perp}$ base by unravelling and pruning the concept descriptions that occur in the GCIs.

Model Exploration Model Exploration can be used to obtain a base for the GCIs holding in a finite model, the background model, when only a working model, a connected submodel of the background model, is available. It is an interactive process, following the pattern of Attribute Exploration, where an expert can inspect each GCI after it has been computed. If a GCI is rejected then the expert is asked to provide a counterexample which is added to the working model. This has to happen in such a way that the working model remains a connected submodel of the background model. Model Exploration uses results from the previous chapter. During the exploration process an implicational base for the context induced by the background model i and the set M_i is computed. In order to compute M_i the background model i needs to be known completely. Since this is not the case, the theory of FCA has been adapted. We have shown that an implicational base can be obtained, even if attributes are added during runtime. This implicational base is not necessarily identical with the Duquenne-Guigues Base, and therefore unfortunately the property of minimal cardinality is lost. Model Exploration always terminates and returns a finite base for the $\mathcal{EL}^{\perp}_{\text{gfp}}$-GCIs that hold in the background model. It can be modified such that cyclic concept descriptions only occur on the right-hand sides of the GCIs. It is not possible to completely avoid cyclic GCIs during runtime. Upon termination it is possible to transform the obtained base into an $\mathcal{EL}^{\perp}$-base using the same construction as in the previous chapter.

ABox Exploration In a classic knowledge base completion setting working models are not present. Furthermore, the connectedness condition on the working models can force the expert to add an unnecessarily large amount of data when providing counterexamples. We have presented ABox Exploration as an alternative to Model Exploration, where counterexamples are stored directly in the ontology, or more specifically in the ABox. In order to describe that an individual a is an (explicit) counterexample to a GCI one needs to specify that a belongs to the concept C but does *not* belong to the concept D. We have shown that, although $\mathcal{EL}^{\bot}$ does not allow for negation, counterexamples can be described in an $\mathcal{EL}^{\bot}$-ontology using disjointness statements.

ABox Exploration differs from Model Exploration mainly in two aspects. First, the counterexamples are stored in the ABox. Second, we have introduced minimal possible consequences, which replace model-based most specific concepts in ABox Exploration. We have also proved that minimal possible consequences always exist for acyclic concept descriptions and $\mathcal{EL}^{\bot}$-ontologies. However, we have not been able to present an efficient method to compute them. We have therefore suggested the use of approximated minimal possible consequences. Their computational behaviour is superior to minimal possible consequences, but they can lead to slightly greater expert interaction during the exploration. It remains to be tested, which method behaves better on real-world data.

Bibliography

[ABB+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[Arm74] W.W. Armstrong. Dependency structures of data base relationships. In Jack L. Rosenfeld, editor, *Proc. of the 6th IFIP Congress*, Stockholm, Sweden, 1974. North-Holland.

[Baa95] Franz Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proc. of the 1st Int. KRUSE Symposium*, pages 168–178, 1995.

[Baa03a] Franz Baader. Least common subsumers and most specific concepts in a Description Logic with existential restrictions and terminological cycles. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 319–324. Morgan Kaufmann, 2003.

[Baa03b] Franz Baader. Terminological cycles in a Description Logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 325–330, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

[BBL05a] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369, Edinburgh (UK), 2005. Morgan Kaufmann, Los Altos.

[BBL05b] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proc. of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BD08] Franz Baader and Felix Distel. A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In Raoul Medina and Sergei Obiedkov, editors, *Proc. of the 6th Int. Conf. on Formal Concept Analysis (ICFCA 2008)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 46–61. Springer, 2008.

[BD09] Franz Baader and Felix Distel. Exploring finite models in the Description Logic $\mathcal{EL}_{\text{gfp}}$. In Sébastien Ferré and Sebastian Rudolph, editors, *Proc. of the 7th Int. Conf. on Formal Concept Analysis (ICFCA 2009)*. Springer, 2009.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[BGSS07] Franz Baader, Bernhard Ganter, Ulrike Sattler, and Barış Sertkaya. Completing Description Logic knowledge bases

using Formal Concept Analysis. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*. AAAI Press/The MIT Press, 2007.

[BH95] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.

[BH00] Peter Burmeister and Richard Holzer. On the treatment of incomplete knowledge in formal concept analysis. In B. Ganter and G.W. Mineau, editors, *Proc. of the 8th Int. Conf. on Conceptual Structures (ICCS 2000)*. Springer, 2000.

[Bir93] Garrett Birkhoff. *Lattice theory*, volume 25 of *Colloquium publications*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1993.

[BK06] Franz Baader and Ralf Küsters. Nonstandard inferences in Description Logics: The story so far. In D.M. Gabbay, S.S. Goncharov, and M. Zakharyaschev, editors, *Mathematical Problems from Applied Logic I*, volume 4 of *International Mathematical Series*, pages 1–75. Springer-Verlag, 2006.

[BK10] Mikhail Babin and Sergei Kuznetsov. Recognizing pseudo-intents is coNP-complete. In Marzena Kryszkiewicz and Sergei Obiedkov, editors, *Proc. of the 7th Int. Conf. on Concept Lattices and Their Applications (CLA 2010)*, volume 672. CEUR Workshop Proceedings, 2010.

[BKM99] Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in Description Logics with existential restrictions. In Thomas Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

[BKT02] Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in Description Logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2002)*, pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

[BL84] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI 1984)*, pages 34–37, 1984.

[BL85] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, 1985.

[BLS06] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.

[BM00] Franz Baader and Ralf Molitor. Building and structuring Description Logic knowledge bases using least common subsumers and concept analysis. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues – Proc. of the 8th Int. Conf. on Conceptual Structures (ICCS 2000)*, volume 1867 of *Lecture Notes in Artificial Intelligence*, pages 290–303. Springer-Verlag, 2000.

[BM05] Sebastian Brandt and Jörg Model. Subsumption in $\mathcal{EL}$ w.r.t. hybrid TBoxes. In Ulrich Furbach, editor, *Proc. of the 28th Annual German Conf. on Artificial Intelligence (KI 2005)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2005.

[BNS08] Franz Baader, Novak Novakovic, and Boontawee Suntisrivaraporn. A proof-theoretic subsumption reasoner for hybrid $\mathcal{EL}$-TBoxes. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proc. of the 2008 Int. Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR-WS*, 2008.

[Bra04] Sebastian Brandt. Polynomial time reasoning in a Description Logic with existential restrictions, GCI axioms, and—what else? In Ramon López de Mántaras and Lorenza

Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.

[Bra06] Sebastian Brandt. *Standard and Non-standard reasoning in Description Logics*. Ph.D. dissertation, Institute for Theoretical Computer Science, TU Dresden, Germany, 2006.

[BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[BS04] Franz Baader and Barış Sertkaya. Applying Formal Concept Analysis to Description Logics. In P. Eklund, editor, *Proc. of the 2nd Int. Conf. on Formal Concept Analysis (ICFCA 2004)*, volume 2961 of *Lecture Notes in Computer Science*, pages 261–286, Sydney, Australia, 2004. Springer-Verlag.

[BS07] Julian Bradfield and Colin Stirling. Modal μ-calculi. In Patrick Blackburn, Johan Van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, chapter 12, pages 721–756. Elsevier, 2007.

[BS09] Franz Baader and Barış Sertkaya. Usability issues in Description Logic knowledge base completion. In Sébastien Ferré and Sebastian Rudolph, editors, *Proc. of the 7th Int. Conf. on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Ingelligence*, pages 1–21. Springer Verlag, 2009.

[BT01] Sebastian Brandt and Anni-Yasmin Turhan. Using non-standard inferences in Description Logics — what does it buy me? In *Proc. of the KI 2001 Workshop on Applications of Description Logics (KIDLWS 2001)*, number 44 in CEUR-WS, Vienna, Austria, September 2001. RWTH Aachen.

[Bur91] Peter Burmeister. Merkmalimplikationen bei unvollständigem Wissen. In W. Lex, editor, *Arbeitstagung Begriffsanalyse und künstliche Intelligenz*, pages 15–46, Clausthal-Zellerfeld, 1991.

[DG84] William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.

[Dis09] Felix Distel. Model-based most specific concepts in some Description Logics with value restrictions. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proc. of the 2009 Int. Workshop on Description Logics (DL 2009)*, volume 477 of *CEUR-WS*, 2009.

[Dis10a] Felix Distel. An approach to exploring Description Logic knowledge bases. In Barış Sertkaya and Léonard Kwuida, editors, *Proc. of the 8th Int. Conf. on Formal Concept Analysis (ICFCA 2010)*, volume 5986 of *Lecture Notes in Artificial Intelligence*, pages 209–224. Springer, 2010.

[Dis10b] Felix Distel. Hardness of enumerating pseudo-intents in the lectic order. In Barış Sertkaya and Léonard Kwuida, editors, *Proc. of the 8th Int. Conf. on Formal Concept Analysis (ICFCA 2010)*, volume 5986 of *Lecture Notes in Artificial Intelligence*, pages 124–137. Springer, 2010.

[DP02] Brian A. Davey and Hillary A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 2nd edition, 2002.

[DS11] Felix Distel and Barış Sertkaya. On the complexity of enumerating pseudo-intents. *Discrete Applied Mathematics*, to appear, 2011.

[EKMS93] M. Erné, J. Koslowski, A. Melton, and G. E. Strecker. A primer on galois connections. *Annals of the New York Academy of Sciences*, 704(1):103–125, 1993.

[Fer02] Sébastien Ferré. *Systèmes d'information logiques: un paradigme logico-contextuel pour interroger, naviguer et apprendre*. PhD thesis, IRISA, France, 2002.

[FR00] Sébastien Ferré and Olivier Ridoux. A logical generalization of Formal Concept Analysis. In Bernhard Ganter and

Guy W. Mineau, editors, *Proc. of the 8th Int. Conf. on Conceptual Structures (ICCS 2000)*, volume 1867 of *LNCS*, pages 371–384. Springer, 2000.

[FR04] Sébastien Ferré and Olivier Ridoux. Introduction to logical information systems. *Information Processing & Management*, 40(3):383 – 419, 2004.

[Gan84] Bernhard Ganter. Two basic algorithms in concept analysis. Preprint 831, Fachbereich Mathematik, TU Darmstadt, Darmstadt, Germany, 1984.

[Gan99] Bernhard Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.

[GD86] J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Humaines*, 95:5–18, 1986.

[GHKS07] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: Extracting modules from ontologies. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proc. of the 16th Int. Conf. on World Wide Web (WWW 2007)*, pages 717–726, Banff, Canada, 2007. ACM.

[GMA95] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[GW97] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, New York, 1997.

[GW99] Bernhard Ganter and Rudolf Wille. Contextual attribute logic. In William M. Tepfenhart and Walling R. Cyre, editors, *Proc. of the 7th Int. Conf. on Conceptual Structures (ICCS 1999)*, pages 377–388, London, UK, 1999. Springer.

[HHK95] M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. of*

the 36th Annual Symposium on Foundations of Computer Science (FOCS 1995), page 453, Washington, DC, USA, 1995. IEEE Computer Society.

[HM01] Volker Haarslev and Ralf Möller. RACER system description. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[Hor98] Ian Horrocks. Using an expressive Description Logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 1998)*, pages 636–647, 1998.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HS04] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, December 2004.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive Description Logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 1999)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[HST00] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–264, 2000.

[KO02] Sergei O. Kuznetsov and Sergei A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14:189–216, 2002.

[KO08] Sergei O. Kuznetsov and Sergei Obiedkov. Some decision and counting problems of the Duquenne-Guigues basis of implications. *Discrete Applied Mathematics*, 156(11):1994 – 2003, 2008. In Memory of Leonid Khachiyan (1952 - 2005).

[Kuz93] Sergei O. Kuznetsov. A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automated Documentation and Mathematical Linguistics*, 27(5):11–21, 1993.

[Kuz04] Sergei O. Kuznetsov. On the intractability of computing the Duquenne-Guigues base. *Journal of Universal Computer Science*, 10(8):927–933, 2004.

[Kuz06] Sergei O. Kuznetsov. Counting pseudo-intents and ♯P-completeness. In Rokia Missaoui and Jürg Schmid, editors, *Proc. of the 4th Int. Conf. on Formal Concept Analysis (ICFCA 2006)*, volume 3874 of *LNCS*, pages 306–308, 2006.

[LPW10a] Carsten Lutz, Robert Piro, and Frank Wolter. Enriching $\mathcal{EL}$-concepts with greatest fixpoints. In R. Studer H. Coelho and M. Wooldridge, editors, *Proc. of the 19th Eur. Conf. on Artificial Intelligence (ECAI 2010)*. IOS Press, 2010.

[LPW10b] Carsten Lutz, Robert Piro, and Frank Wolter. $\mathcal{EL}$-concepts go second order: Greatest fixpoints and simulation quantifiers. In *Proc. of the 23rd Int. Workshop on Description Logics (DL 2010)*, volume 573 of *CEUR-WS*, 2010.

[Lut03] Carsten Lutz. Description Logics with concrete domains—a survey. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.

[Min81] Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*. The MIT Press, 1981. A longer version appeared in *The Psychology of Computer Vision* (1975). Republished in [BL85].

[MS09] Julian Mendez and Boontawee Suntisrivaraporn. Reintroducing CEL as an OWL 2 EL reasoner. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proc. of the 2009 Int. Workshop on Description Logics (DL 2009)*, volume 477 of *CEUR-WS*, 2009.

[Neb88] Bernhard Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[Neb91] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[NR99] Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71(5-6):199 – 204, 1999.

[Peñ09] Rafael Peñaloza. *Axiom-Pinpointing in Description Logics and Beyond*. PhD thesis, TU Dresden, Dresden, Germany, 2009.

[Pre97] Susanne Prediger. Logical scaling in Formal Concept Analysis. In Dickson Lukose, Harry S. Delugach, Mary Keeler, Leroy Searle, and John F. Sowa, editors, *Proc. of the 5th Int. Conf. on Conceptual Structures (ICCS 1997)*, volume 1257 of *LNCS*, pages 332–341. Springer, 1997.

[Pre00] Susanne Prediger. Terminologische Merkmalslogik in der Formalen Begriffsanalyse. In Gerd Stumme and Rudolf Wille, editors, *Begriffliche Wissensverarbeitung: Methoden und Anwendungen*, pages 99–124, Berlin, Heidelberg, New York, 2000. Springer.

[Pri98] Uta Priss. The formalization of WordNet by methods of relational concept analysis. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database and some of its applications*, Cambridge, Massachusetts, 1998. MIT Press.

[PS99] Susanne Prediger and Gerd Stumme. Theory-driven logical scaling: Conceptual information systems meet Description Logics. In *Knowledge Representation Meets Databases*, pages 46–49, 1999.

[PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In Allan Ellis and Tatsuya Hagino, editors, *Proc. of the 14th International Conference on World Wide Web (WWW'05)*, pages 633–640. ACM, 2005.

[PW99] Susanne Prediger and Rudolf Wille. The lattice of concept graphs of a relationally scaled context. In William M. Tepfenhart and Walling R. Cyre, editors, *Proc. of the 7th Int. Conf. on Conceptual Structures (ICCS 1999)*, volume 1640 of *Lecture Notes in Computer Science*, pages 401–414. Springer, 1999.

[RH97] Alan Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a Description Logic with transitivity and concept inclusions. In *Proc. of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI 1997)*, Stanford, CA, 1997. AAAI Press.

[Rud04] Sebastian Rudolph. Exploring relational structures via $\mathcal{FLE}$. In K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, editors, *Proc. of the 12th Int. Conf. on Conceptual Structures (ICCS 2004)*, volume 3127 of *LNCS*, pages 196–212, 2004.

[Rud06] Sebastian Rudolph. *Relational Exploration – Combining Description Logics and Formal Concept Analysis for Knowledge Specification.* PhD thesis, Technische Universität Dresden, 2006.

[RVH07] Sebastian Rudolph, Johanna Völker, and Pascal Hitzler. Supporting lexical ontology learning by relational exploration. In Uta Priss, Simon Polovina, and Richard Hill, editors, *Proc. of the 15th Int. Conf. on Conceptual Structures (ICCS 2007): Knowledge Architectures for Smart Applications*, pages 488–491, Berlin, Heidelberg, 2007. Springer-Verlag.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

[SCC97] K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.

[Sch91] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991)*, pages 466–471, 1991.

[Ser07] Barış Sertkaya. *Formal Concept Analysis Methods for Description Logics*. Ph.D. thesis, TU Dresden, Dresden, Germany, 2007.

[Ser08] Barış Sertkaya. Explaining user errors in Description Logic knowledge base completion. In *Inf. Proc. of the 2008 Int. Workshop on Complexity, Expressibility, and Decidability in Automated Reasoning (CEDAR 2008)*, 2008.

[Ser09a] Barış Sertkaya. OntoComP: A protegé plugin for completing OWL ontologies. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Paslaru Bontas Simperl, editors, *Proc. of the 6th Eur. Semantic Web Conference, (ESWC 2009)*, volume 5554 of *Lecture Notes in Computer Science*, pages 898–902. Springer Verlag, 2009.

[Ser09b] Barış Sertkaya. Some computational problems related to pseudo-intents. In Sébastien Ferré and Sebastian Rudolph, editors, *Proc. of the 7th Int. Conf. on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Intelligence*, pages 130–145. Springer Verlag, 2009.

[Sow91] John F. Sowa, editor. *Principles of Semantic Networks.* Morgan Kaufmann, Los Altos, 1991.

[Sow92] John F. Sowa. Semantic networks. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence (2nd Edition).* John Wiley & Sons, 1992.

[STB+00] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Fast computation of concept lattices using data mining techniques. In *Proc. 7th Int. Workshop on Knowledge Representation Meets Databases (KRDB 2000)*, pages 129–139, 2000.

[Stu96a] Gerd Stumme. Attribute exploration with background implications and exceptions. In H.-H. Bock and W. Polasek, editors, *Data Analysis and Information Systems*, page 457ff, Berlin, 1996. Springer.

[Stu96b] Gerd Stumme. The concept classification of a terminology extended by conjunction and disjunction. In N. Foo and R. Goebel, editors, *PRICAI'96: Topics in Artificial Intelligence. Proc. PRICAI'96*, volume 1114 of *LNAI*, pages 121–131, Heidelberg, 1996. Springer.

[Stu96c] Gerd Stumme. Exploration tools in Formal Concept Analysis. In E. Diday, Y. Lechevallier, and O. Opitz, editors, *Proc. of the Int. Conf. on Ordinal and Symbolic Data Analysis (OSDA 1995)*, Studies in classification, data analysis, and knowledge organization, pages 31–44, Berlin–Heidelberg, 1996. Springer–Verlag.

[Stu02] Gerd Stumme. Efficient data mining based on Formal Concept Analysis. In Abdelkader Hameurlain, Rosine Cicchetti, and Roland Traunmüller, editors, *Database and Expert Systems Applications*, volume 2453 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2002.

[Sun09] Boontawee Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.* PhD thesis, TU Dresden, 2009.

[TH06] Dmitry Tsarkov and Ian Horrocks. Fact++ Description Logic reasoner: System description. In Natarajan Shankar Ulrich Furbach, editor, *In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, pages 292–297. Springer, 2006.

[Tur07] Anni-Yasmin Turhan. *On the Computation of Common Subsumers in Description Logics*. PhD thesis, TU Dresden, Dresden, Germany, 2007.

[WGG08] Johannes Wollbold, Reinhard Guthke, and Bernhard Ganter. Constructing a knowledge base for gene regulatory dynamics by Formal Concept Analysis methods. In Katsuhisa Horimoto, Georg Regensburger, Markus Rosenkranz, and Hiroshi Yoshida, editors, *Algebraic Biology*, volume 5147 of *LNCS*. Springer, 2008.

[Wil82] Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered Sets: Proc. of the NATO Advanced Study Institute*, pages 445–470. Reidel, 1982.

[Zic91] Monika Zickwolff. *Rule Exploration: First Order Logic in Formal Concept Analysis*. PhD thesis, TH Darmstadt, Germany, 1991.