Technische Universität Dresden

Faculty of Computer Science
Institute of Theoretical Computer Science, Chair of Automata Theory

Master's Thesis

on

Complexity of Matching in the Description Logic $\mathcal{EL}$ without the Top concept

by

Puneetha Jangir Lok Ram Jangir
born on 5 February 1997 in Rajasthan, India

Examiner: **Dr.-Ing. Stefan Borgwardt**
Supervisor: **Dr. Oliver Fernández Gil**

October 9, 2024

Task for the preparation of a master's thesis

Author: **Puneetha Jangir Lok Ram Jangir**
Matriculation Number: **5050581**
Degree: **Master of Science**
Course: **Computational Modeling and Simulation, Logic Modeling**
Title: **Complexity of Matching in the Description Logic $\mathcal{EL}$ without the Top concept**
Supervisor: **Dr. Oliver Fernández Gil**
Date of Submission: **9 October 2024**

Objectives of the work

- Review and study the existing literature about matching and unification in the DL $\mathcal{EL}$.
- Determine the complexity of different variants of the matching problem in the DL $\mathcal{EL}^{-\top}$.
- Support the correctness of the new results by providing corresponding algorithms and formal proofs.

Declaration of Authorship

I hereby certify that I have authored this document entitled **Complexity of Matching in the Description Logic $\mathcal{EL}$ without the Top concept** independently and without undue assistance from third parties. No other than the resources and references indicated in this document have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present document. I am aware that violations of this declaration may lead to subsequent withdrawal of the academic degree.

October 9, 2024
Puneetha Jangir Lok Ram Jangir

Puneetha Jangir
Lok Ram Jangir

Digitally signed by Puneetha Jangir Lok Ram Jangir
Date: 2024.10.09 22:19:12 +02'00'

Abstract

Matching and unification have been proposed as reasoning problems in description logics that can, for example, be used to detect redundancies or filter out unimportant aspects of large concept descriptions in ontologies such as SNOMED CT [1]. In the particular case of matching in $\mathcal{EL}$, the complexity of the problem was shown to be NP-complete in [BK00] and [BM14a]. However, the problem has not been investigated for the DL $\mathcal{EL}^{-\top}$, which is a fragment of $\mathcal{EL}$ that does not allow the use of the $\top$ concept. In this thesis, we will explore the computational complexity of matching in $\mathcal{EL}^{-\top}$. We will consider different variants of the matching problem in $\mathcal{EL}^{-\top}$, both in the case of an empty TBox and in the presence of general $\mathcal{EL}^{-\top}$-TBoxes. In particular, we will prove that the complexity of matching w.r.t. general $\mathcal{EL}^{-\top}$-TBoxes increases if the $\top$ concept is not allowed. More precisely, we show that general matching in $\mathcal{EL}^{-\top}$ w.r.t. arbitrary $\mathcal{EL}^{-\top}$-TBoxes is a PSpace-hard problem and that it can be decided in exponential time. The complexity results obtained in this thesis for all variants of the matching problem in $\mathcal{EL}^{-\top}$ are summarized in Table 6.2.

[1] http://www.ihtsdo.org/snomed-ct

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Oliver Fernández Gil, for his continuous support, guidance and endless patience throughout the course of my thesis. I am very much thankful to him for giving me the opportunity to work on this thesis as it has broadened my knowledge in description logics to a great extent.

I would also like to sincerely thank Dr.-Ing. Stefan Borgwardt for agreeing to review my thesis despite his tight schedule.

This thesis would not have been possible without the constant and unwavering support, care and encouragement from my friends and family, for which I am profoundly grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Description Logics

Description Logics (DLs) [Baa+03] are a family of knowledge representation languages that have been extensively studied for their ability to represent knowledge in a structured and precise manner. They serve as the foundation for various applications within artificial intelligence, enabling the representation, maintenance and reasoning of knowledge across diverse domains. By using DLs, knowledge can be encoded in a way that is both human-understandable and machine-processable. This structured representation facilitates the development of applications that can perform sophisticated reasoning to derive new knowledge, validate existing information, and ensure consistency within knowledge bases. A few applications include, (1) **Ontology development**: They are widely used in ontology languages like OWL (Web Ontology Language) [1], enabling the creation of complex ontologies that underpin the semantic web. These ontologies support data inter-operability and semantic search by providing a shared understanding of domain concepts. (2) **Medical Domain**: They are used to represent and reason about big biomedical ontologies like SNOMED CT or the Gene ontology [2].

DLs are a fragment of first-order logic and *concepts* in DLs are the building blocks used to represent specific knowledge within the domain of interest. They are constructed using *concept names* that represent

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

basic categories (e.g., Human, Female) and *role names* that represent relationships between individuals (e.g., hasChild). Concept constructors are the operators that combine concept names and role names to form more complex concepts. The meaning or semantics of a concept term is defined through *interpretations* $\mathcal{I}$, also inherited from first-order logic. An interpretation consists of a non-empty set of individuals $\Delta^{\mathcal{I}}$, that represent the entities within the domain being described (e.g., all people in the world) and an interpretation function $\cdot^{\mathcal{I}}$, that assigns meaning to concept names and role names. Concept names are interpreted as subsets of the domain (e.g., Human might represent all humans) and role names are interpreted as binary relations between individuals (e.g., hasChild relates a parent to its children). For example, consider the concept of "women having daughters." We can represent this using the following concept term,

$$Human \sqcap Female \sqcap \exists hasChild.Female.$$

The description above specifies the intersection of being a human, a female, and having at least one female child.

The basic building blocks can be combined to define more complex concepts by making use of concept (or logical) constructors such as $\sqcap, \sqcup, \neg, \exists$ or $\forall$. Besides the construction of complex concepts, DLs also offer the possibility to define ontological knowledge, i.e., to formulate ontologies. One way to do so is using a TBox introduced in the Section 2.1.

DLs vary in their expressive power, which depends on the range of constructors they support and the types of axioms that can be expressed. There is often a trade-off between expressivity and the complexity of reasoning tasks. Highly expressive DLs may allow the representation of complex knowledge but at the cost of increased reasoning complexity, which can be computationally expensive. Conversely, DLs with limited expressivity like the DL $\mathcal{EL}$, often enable more efficient reasoning, making them suitable for applications where complex constructs are unnecessary.

The DL $\mathcal{EL}$ is a member of the family of description logics (DLs) and is particularly notable for its simplicity and efficiency in reasoning tasks. The DL $\mathcal{EL}$ provides the concept constructors, conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). Despite its limited

expressivity, the bio-medical ontologies mentioned above like SNOMED CT, make significant use of the polynomial time reasoning capabilities that $\mathcal{EL}$ provides [BKM99].

One of the fundamental reasoning tasks in DLs is subsumption checking. A concept C is subsumed by another concept D (written as, $C \sqsubseteq D$), if in every interpretation, the set of individuals satisfying C is always a subset of the set of individuals satisfying D. Intuitively, this means that C is a more specific concept than D. For example, the following subsumption holds,

$$Human \sqcap Female \sqcap \exists hasChild.Female \sqsubseteq Human \sqcap Female.$$

It is because all women having daughters are also necessarily women. Subsumption checking has shown to be decidable in polynomial time for the description logic $\mathcal{EL}$, even in the presence of arbitrary TBoxes [BBL05]. Subsumption checking allows to build a hierarchy of concepts, where more specific concepts are subsumed by more general ones. We say two concepts C and D are equivalent written as $C \equiv D$, if they subsume each other, i.e., $C \sqsubseteq D$ and $D \sqsubseteq C$. Such an equivalence check helps identify redundancies within a knowledge base.

1.2 Matching in Description Logics

Matching or in a more general form, *Unification* in DLs is a process of finding correspondences between concept descriptions and concept patterns [BN01]. It serves several purposes, such as **filtering out** unimportant aspects of large concept descriptions [BM96], **detecting redundancies** in knowledge bases [BN01], and aiding in the integration of knowledge bases. *Concept patterns* are concept descriptions containing variables. For example, consider a scenario where we want to identify concepts involving individuals who have both a son and a daughter sharing a particular characteristic. This can be represented by the pattern,

$$D := \exists \text{has-child.(Male} \sqcap X) \sqcap \exists \text{has-child(Female} \sqcap X),$$

where X is a variable denoting the shared characteristic. For instance, the concept description $C := \exists$has-child(Tall $\sqcap$ Male) $\sqcap \exists$has-child.(Tall $\sqcap$Female) fits this pattern. By substituting X with Tall the pattern D becomes equivalent to C. Therefore, the substitution $\sigma := \{X \mapsto \text{Tall}\}$

is a matcher modulo equivalence for the matching problem $C \equiv^? D$, since $C \equiv \sigma(D)$. However, the paper [BM96] focused on matching modulo subsumption rather than equivalence. In this context, the problem is formulated as $C \sqsubseteq^? D$, and a matcher is a substitution σ that satisfies $C \sqsubseteq \sigma(D)$. While any matcher modulo equivalence is also a matcher modulo subsumption, the converse is not true. For example, the substitution $\sigma_\tau := \{X \mapsto \top\}$ is a matcher modulo subsumption for the problem $C \sqsubseteq^? D$, but it is not a matcher modulo equivalence for $C \equiv^? D$.

The DL $\mathcal{EL}^{-\top}$ consists of all concept constructors as defined for $\mathcal{EL}$ with the exception that the top concept ($\top$) is not allowed. The semantics remain the same as for $\mathcal{EL}$. Considering the example above, the concept pattern X cannot be substituted with $\top$ as it is not allowed. The substitution $\sigma := \{X \mapsto \text{Tall}\}$ is a valid $\mathcal{EL}^{-\top}$ matcher of C and D. The matcher(s) modulo equivalence or subsumption now must be more specific which makes the problem more computationally challenging. Exploring matching in $\mathcal{EL}^{-\top}$ is an interesting problem as SNOMED CT is formulated in $\mathcal{EL}^{-\top}$. For detecting redundancies in ontologies like SNOMED CT, matching in $\mathcal{EL}$ would introduce concept terms containing the $\top$ concept, whereas the only one top concept in SNOMED is the SNOMED CT concept. It is used as the root of a concept hierarchy. Having another $\top$ concept would make this concept redundant. Therefore, we explore the computational complexity of matching in the DL $\mathcal{EL}^{-\top}$.

Different variants of matching in $\mathcal{EL}$ were introduced in [BM14a]. Matching modulo subsumption, specifically left-ground and right-ground matching modulo subsumption was shown to be in PTime. Matching modulo equivalence w.r.t. an empty TBox was shown to be NP-complete in [BK00]. General matching in $\mathcal{EL}$ is decidable and was also shown to be NP-complete in [BK00; BM14a], even in the presence of arbitrary TBoxes. However, unification in $\mathcal{EL}^{-\top}$ was shown to be PSpace-complete in [Baa+11b].

1.3 Motivation of the thesis

The main goal of this thesis is to decide the solvability of matching in $\mathcal{EL}^{-\top}$. When considering $\mathcal{EL}^{-\top}$, the dynamics of matching undergo a

significant transformation, which we will explore in detail. Consider an $\mathcal{EL}$ matching problem Γ,

$$\Gamma := \{A \sqcap B \sqsubseteq^? X, A \sqsubseteq^? X\},$$

where A and B are concept names and X is a concept variable. In order to find a substitution σ that solves the subsumption constraints in Γ, we need to ensure that σ satisfies $A \sqcap B \sqsubseteq^? \sigma(X)$ and $A \sqsubseteq^? \sigma(X)$. Obviously, the substituion that replaces X by $\top$ is an $\mathcal{EL}$ matcher of Γ. However, Γ does not have an $\mathcal{EL}^{-\top}$ matcher. Every $\mathcal{EL}^{-\top}$ matching problem Γ is also an $\mathcal{EL}$ matching problem and if σ is an $\mathcal{EL}^{-\top}$ matcher of Γ, it is also an $\mathcal{EL}$ matcher of Γ, but the converse is not true. As shown in the example above, Γ does not have an $\mathcal{EL}^{-\top}$ matcher and if it exists there are no methods available to find one. For this reason, it is worthwile to investigate matching in $\mathcal{EL}^{-\top}$. Matching in $\mathcal{EL}^{-\top}$ poses several challenges. The main challenge is the reduced flexibility in pattern formulation due to the absence of $\top$ concept. This specificity required by $\mathcal{EL}^{-\top}$ can lead to more accurate and relevant matchers.

1.4 Structure of the thesis

The thesis is structured as follows,

- In chapter 2, we will formally introduce description logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ along with basic notions related to these logics.
- In chapter 3, we will define the matching problem in $\mathcal{EL}^{-\top}$ and introduce different variants of this problem, whose computational complexities will be explored throughout the thesis.
- In chapter 4, we demonstrate that left-ground matching modulo subsumption in $\mathcal{EL}^{-\top}$ is NP-complete when considering an empty TBox, and becomes PSpace-hard in the presence of general $\mathcal{EL}^{-\top}$-TBoxes. Additionally, we examine the impact of these results on the complexity of matching modulo equivalence. However, we show that the complexity of right-ground matching modulo subsumption in $\mathcal{EL}^{-\top}$ remains in PTime.
- In chapter 5, we prove that general matching in $\mathcal{EL}^{-\top}$ is in ExpTime, using ideas from the goal oriented matching algorithm presented in [BM14a]. We will also present an algorithm to decide

the existence of common subsumers without $\top$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox in this chapter.

- In chapter 6, we summarize the complexity results of the different matching problems presented in the thesis and propose some directions for future work.

Chapter 2

The Description Logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

In this chapter, we introduce definitions that will be used in the subsequent chapters. We start by formally introducing the syntax and semantics for the logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$. Basic notions related to these logics are also presented. Next, we define characterization of subsumption in $\mathcal{EL}$ and state some properties of subsumption in this logic. Finally, we define canonical models, simulation relations and related notions that will be used in the algorithms presented in chapter 5.

2.1 Syntax and Semantics

The syntax is based on a finite set of concept names N_C and role names N_R. $\mathcal{EL}$ concept descriptions are built using the constructors conjunction ($C \sqcap D$), existential restriction ($\exists r.C$) and top ($\top$). The set of $\mathcal{EL}$ concept descriptions is the smallest set satisfying the conditions below:

- A is an $\mathcal{EL}$ concept description, for all $A \in N_C$.
- If C, D are $\mathcal{EL}$ concept descriptions, then $C \sqcap D$ is also an $\mathcal{EL}$ concept description.
- If C is an $\mathcal{EL}$ concept description and $r \in N_R$ a role name, then $\exists r.C$ is also an $\mathcal{EL}$ concept description.

The DL $\mathcal{EL}^{-\top}$ is fragment of $\mathcal{EL}$ where the use of top-concept $\top$ is not allowed, i.e, concept descriptions in $\mathcal{EL}^{-\top}$ are $\mathcal{EL}$ concepts that

do not contain $\top$. Most of the definitions that follow are given for $\mathcal{EL}$, otherwise it is explicitly stated that it is for $\mathcal{EL}^{-\top}$.

The semantics of $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ are defined using standard first-order logic interpretations. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, consists of a non-empty domain $\Delta^{\mathcal{I}}$, and an interpretation function $\cdot^{\mathcal{I}}$ that assigns subsets of $\Delta^{\mathcal{I}}$ to each concept name and binary relations over $\Delta^{\mathcal{I}}$ to each role name. For the semantics of complex concept descriptions, refer Table 2.1.

Let C and D be two $\mathcal{EL}$ concepts. A *general concept inclusion* (GCI) is an expression of the form $C \sqsubseteq D$. A general $\mathcal{EL}$-TBox is a finite set of such GCIs. We say that $C \sqsubseteq D$ is satisfied in an interpretation $\mathcal{I}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it satisfies all GCIs in $\mathcal{T}$.

Name	**Syntax**	**Semantics**
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$

Table 2.1: Syntax and Semantics of $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

Next, we will define the notion of atoms and particles. For a concept description C, we will also define the notion of sub-concepts and role-depth.

Definition 2.1.1. An $\mathcal{EL}$ concept description is called an *atom* if it is a concept name or an existential restriction. The set $At(C)$ of all atoms of an $\mathcal{EL}$ concept description C is defined as follows:

- If C is $\top$, then $At(C) := \emptyset$
- If C is a concept name, then $At(C) := \{C\}$.
- If $C = \exists r.D$, then $At(C) := \{C\} \cup At(D)$.
- If $C = C_1 \sqcap C_2$, then $At(C) := At(C_1) \cup At(C_2)$.

Flat atoms are concept names A or existential restrictions $\exists r.D$, where D is a concept name or $\top$. Note that in $\mathcal{EL}^{-\top}$, the concept D cannot be $\top$. A concept description is *flat* if it is a conjunction of flat atoms.

Every $\mathcal{EL}$ concept description C is defined as a conjunction of atoms $C_1, \ldots, C_n$, with $n \geq 0$, which are called the top level atoms of C. The case where $n = 0$ yields an empty conjunction that corresponds to the $\top$ concept. However, for $\mathcal{EL}^{-\top}$ the top level atoms of C are defined as *particles*, as the $\top$ concept is not allowed.

Definition 2.1.2. Particles are concept terms of the form $\exists r_1.\exists r_2. \ldots \exists r_n.A$ (abbreviated as $\exists r_1 \ldots r_n.A$) for role names $r_1, \ldots, r_n$ and a concept name A. The set $Part(C)$ of all particles of an $\mathcal{EL}^{-\top}$-concept term C is defined as follows:

- If C is a concept name, $Part(C) := \{C\}$.
- If $C = \exists r.D$, then $Part(C) := \{\exists r.M \mid M \in Part(D)\}$.
- If $C = C_1 \sqcap C_2$, then $Part(C) := Part(C_1) \cup Part(C_2)$.

For example, the particles of the concept $A \sqcap \exists r.(A \sqcap \exists r.(A \sqcap B))$ where $A, B \in N_C$ and $r \in N_R$, are A, $\exists r.A$, $\exists rr.A$ and $\exists rr.B$.

Definition 2.1.3. [BF16] Let C be an $\mathcal{EL}$ concept description. The set $sub(C)$ of sub-concepts of C is defined as follows:

- If $C = \top$ or $C \in N_C$, then $sub(C) = \{C\}$.
- If $C = C_1 \sqcap C_2$, then $sub(C) = \{C\} \cup sub(C_1) \cup sub(C_2)$.
- If $C = \exists r.D$, then $sub(C) = \{C\} \cup sub(D)$.

Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox, for all GCIs in $\mathcal{T}$ the set $sub(\mathcal{T})$ can be analogously defined as follows,

$$sub(\mathcal{T}) := \bigcup_{C \sqsubseteq D \in \mathcal{T}} (sub(C) \cup sub(D)).$$

Definition 2.1.4. [BF16] For a concept description C, the role-depth $rd(C)$ of C is defined as follows:

- If $C = \top$ or $C \in N_C$, then $rd(C) = 0$.
- If $C = \exists r.D$, then $rd(C) = rd(D) + 1$.

- If $C = C_1 \sqcap C_2$, then $rd(C) = max\{rd(C_1), rd(C_2)\}$.

Subsumption and equivalence relations between two $\mathcal{EL}$ concepts w.r.t. a general $\mathcal{EL}$-TBox $\mathcal{T}$ are defined below.

An $\mathcal{EL}$ concept description C is subsumed by another $\mathcal{EL}$ concept description D w.r.t. a general $\mathcal{EL}$-TBox $\mathcal{T}$ (written as $C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all models $\mathcal{I}$ of $\mathcal{T}$. The concept C is equivalent to D w.r.t. $\mathcal{T}$ (written as $C \equiv_{\mathcal{T}} D$) iff $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$ in every model $\mathcal{I}$ of $\mathcal{T}$. If $\mathcal{T}$ is empty, we write $C \sqsubseteq D$ and $C \equiv D$ instead of $C \sqsubseteq_{\mathcal{T}} D$ and $C \equiv_{\mathcal{T}} D$.

2.2 Characterization of subsumption

In this section, we define the characterization of subsumption for both the logics $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$. Based on these characterizations, some properties about the subsumption relation in the considered logics are provided. These properties will also be used in algorithms presented in Chapter 5 for matching in $\mathcal{EL}^{-\top}$.

The characterization of subsumption w.r.t. an empty TBox is defined below.

Lemma 2.2.1. *[BM10] Let $C = A_1 \sqcap \ldots \sqcap A_k \sqcap \exists r_1.C_1 \sqcap \ldots \sqcap \exists r_m.C_m$ and $D = B_1 \sqcap \ldots \sqcap B_l \sqcap \exists s_1.D_1 \sqcap \ldots \sqcap \exists s_n.D_n$ be two $\mathcal{EL}$-concept terms, where $A_1, \ldots, A_k$,$B_1, \ldots, B_l$ are concept names. Then $C \sqsubseteq D$ iff $\{B_1, \ldots, B_l\} \subseteq \{A_1, \ldots, A_k\}$ and for every $j \in \{1, \ldots, n\}$ there exists an $i \in \{1, \ldots, m\}$ such that $r_i = s_j$ and $C_i \sqsubseteq D_j$.*

As stated in [Baa+11b], the characterization of subsumption in $\mathcal{EL}$ also holds for $\mathcal{EL}^{-\top}$-concept descriptions.

The consequence of the characterization in Lemma 2.2.1 is the following lemma.

Lemma 2.2.2. *[BM10] Let C be an $\mathcal{EL}^{-\top}$-concept term and B a particle.*

1. *If $B \sqsubseteq C$, then $B \equiv C$.*
2. *$B \in Part(C)$ iff $C \sqsubseteq B$.*

The lemma above states a property of subsumption that will be useful later on.

The characterization of subsumption w.r.t. a general $\mathcal{EL}$-TBox $\mathcal{T}$ is presented next. As defined for $\mathcal{EL}$ in [BBM12], subsumption between two atoms in $\mathcal{EL}$ is *structural* if their top-level structure is compatible. More precisely, consider two $\mathcal{EL}^{-\top}$-atoms C and D, we say that C is *structurally subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq^s_{\mathcal{T}} D$) iff the following holds,

1. $C = D$ is a concept name, or
2. $C = \exists r.C'$, $D = \exists r.D'$, and $C' \sqsubseteq_{\mathcal{T}} D'$.

The characterization of subsumption presented above also holds w.r.t. a general $\mathcal{EL}^{-\top}$-TBox [Baa+11b]. This characterization will also be used in the algorithms presented in Chapter 5 for matching in $\mathcal{EL}^{-\top}$.

Lemma 2.2.3. *[BBM12] Let $\mathcal{T}$ be an $\mathcal{EL}$-TBox and $C_1, ..., C_n$, $D_1, ..., D_m$ be $\mathcal{EL}$ atoms. Then $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_m$ holds iff for every $j \in \{1, \ldots, m\}$,*

1. *there is an index $i \in \{1, \ldots, n\}$ such that $C_i \sqsubseteq^s_{\mathcal{T}} D_j$ or*
2. *there are atoms $A_1, \ldots, A_k$, B of $\mathcal{T}$ ($k \geq 0$) such that*
   (a) *$A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$,*
   (b) *for every $\eta \in \{1, \ldots, k\}$ there is $i \in \{1, \ldots, n\}$ with $C_i \sqsubseteq^s_{\mathcal{T}} A_\eta$, and*
   (c) *$B \sqsubseteq^s_{\mathcal{T}} D_j$*

An alternative characterization of subsumption can be stated in terms of canonical models, as described in [ZT13a; LW10].

Lemma 2.2.4. *[LW10] Let C and D be two $\mathcal{EL}$ concepts and $\mathcal{T}$ be an $\mathcal{EL}$-TBox. The following conditions are equivalent:*

1. *$C \sqsubseteq_{\mathcal{T}} D$.*
2. *$d_C \in D^{\mathcal{I}_{C,\mathcal{T}}}$.*

The property of characteristic concepts as shown below in Lemma 2.2.5 will be used in the matching algorithms for $\mathcal{EL}^{-\top}$ in chapter 5.

Lemma 2.2.5. *[ZT13a] Let $(\mathcal{I}, d)$ and $(\mathcal{J}, e)$ be interpretations. Then $e \in (X^\ell(\mathcal{I}, d))^{\mathcal{J}}$ iff $(\mathcal{I}^\ell_d, d) \lesssim (\mathcal{J}, e)$.*

2.3 Canonical models and Simulation relations

The purpose of this section is to introduce notions that are crucial to the algorithms presented in chapter 5 to decide matching in $\mathcal{EL}^{-\top}$.

We start by defining the canonical model of an $\mathcal{EL}$ concept C and and an $\mathcal{EL}$-TBox $\mathcal{T}$ as shown below.

Definition 2.3.1. [LW10] Let C be an $\mathcal{EL}$ concept and $\mathcal{T}$ be an $\mathcal{EL}$-TBox. The canonical model $\mathcal{I}_{C,\mathcal{T}}$ of C and $\mathcal{T}$ is defined as:

- $\Delta^{\mathcal{I}_{C,\mathcal{T}}} := \{d_C\} \cup \{d_{C'} \mid \exists r.C' \in sub(C) \cup sub(\mathcal{T})\}$;
- $A^{\mathcal{I}_{C,\mathcal{T}}} := \{d_D \mid D \sqsubseteq_{\mathcal{T}} A\}$, for all $A \in N_C$;
- $r^{\mathcal{I}_{C,\mathcal{T}}} := \{(d_D, d_{D'}) \mid D \sqsubseteq_{\mathcal{T}} \exists r.D'$ for $\exists r.D' \in sub(\mathcal{T})$ or $\exists r.D'$ is a conjunct in $D\}$, for all $r \in N_R$.

Simulation relations between interpretations can be used to identify some properties of the canonical models.

Operations on these interpretations include tree unraveling of the interpretation and construction of characteristic concepts.

Definition 2.3.2. [LW10] Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations and $\mathcal{S} \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$. Then, $\mathcal{S}$ is called a simulation from $\mathcal{I}_1$ to $\mathcal{I}_2$ if the following conditions are satisfied:

- For all concept names $A \in N_C$ and all $(e_1, e_2) \in \mathcal{S}$ it holds: $e_1 \in A^{\mathcal{I}_1}$ implies $e_2 \in A^{\mathcal{I}_2}$.
- For all role names $r \in N_R$ and all $(e_1, e_2) \in \mathcal{S}$, and all $f_1 \in \Delta^{\mathcal{I}_1}$ with $(e_1, f_1) \in r^{\mathcal{I}_1}$, there exists $f_2 \in \Delta^{\mathcal{I}_2}$ such that $(e_2, f_2) \in r^{\mathcal{I}_2}$ and $(f_1, f_2) \in \mathcal{S}$.

An interpretation $\mathcal{I}$ with $d \in \Delta^{\mathcal{I}}$ is denoted as $(\mathcal{I}, d)$. If there exists a simulation $\mathcal{S} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ with $(d, e) \in \mathcal{S}$ then we say that $(\mathcal{I}, d)$ is simulated by $(\mathcal{J}, e)$ denoted as $(\mathcal{I}, d) \lesssim (\mathcal{J}, e)$.

The product of two interpretations is defined as follows.

Definition 2.3.3. Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations. The product inter-

pretation $\mathcal{I} \times \mathcal{J}$ is defined by

$$\Delta^{\mathcal{I}\times\mathcal{J}} := \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}};$$
$$A^{\mathcal{I}\times\mathcal{J}} := \{(d,e) \mid (d,e) \in \Delta^{\mathcal{I}\times\mathcal{J}} \wedge d \in A^{\mathcal{I}} \wedge e \in A^{\mathcal{J}}\}, \text{for all } A \in N_C;$$
$$r^{\mathcal{I}\times\mathcal{J}} := \{((d,e),(f,g)) \mid ((d,e),(f,g)) \in \Delta^{\mathcal{I}\times\mathcal{J}} \times \Delta^{\mathcal{I}\times\mathcal{J}}$$
$$\wedge (d,f) \in r^{\mathcal{I}} \wedge (e,g) \in r^{\mathcal{J}}\}, \text{for all } r \in N_R.$$

Given an interpretation $\mathcal{I}$ and an element d of the domain $\Delta^{\mathcal{I}}$. The interpretation can be unravelled into a possibly infinite tree with root d. The nodes of this tree are words that correspond to a path in $\mathcal{I}$ starting from d. Now, $\pi = dr_1d_1r_2d_2r_3\cdots$ is a path in an interpretation $\mathcal{I}$ if the domain elements d_i and d_{i+1} are connected via $r^{\mathcal{I}}_{i+1}$ for all $1 \leq i \leq \mathbb{N}$.

Definition 2.3.4. [ZT13a] Let $\mathcal{I}$ be an interpretation with $d \in \Delta^{\mathcal{I}}$. The tree unraveling $\mathcal{I}_d$ of $\mathcal{I}$ in d is defined as follows:

$$\Delta^{\mathcal{I}_d} := \{dr_1d_1r_2\cdots r_nd_n \mid (d_i,d_{i+1}) \in r^{\mathcal{I}}_{i+1} \wedge 0 \leq i < n \wedge d_0 = d \wedge n \geq 0\};$$
$$A^{\mathcal{I}_d} := \{\sigma d' \mid \sigma d' \in \Delta^{\mathcal{I}_d} \wedge d' \in A^{\mathcal{I}}\}, \text{for all } A \in N_C;$$
$$r^{\mathcal{I}_d} := \{(\sigma, \sigma r d') \mid (\sigma, \sigma r d') \in \Delta^{\mathcal{I}_d} \times \Delta^{\mathcal{I}_d}\}, \text{for all } r \in N_R.$$

The length of an element $\tau \in \Delta^{\mathcal{I}_d}$, denoted by $|\tau|$, is the number of role names occurring in τ. If τ is of the form $dr_1d_1r_2\cdots r_md_m$, then d_m is the tail of τ denoted by $\text{tail}(\sigma) = d_m$. The interpretation $\mathcal{I}^{\ell}_d$ denotes the finite subtree rooted at d of the tree unraveling $\mathcal{I}_d$ containing all elements up to depth k. Such a finite tree can be translated into a complex concept known as the k-characteristic concept.

Definition 2.3.5. [ZT13b] Let $(\mathcal{I}, d)$ be the tree unraveling $\mathcal{I}_d$ of $\mathcal{I}$ in d. The k-characteristic concept $X^k(\mathcal{I}, d)$ is defined as follows:

$$X^0(\mathcal{I},d) := \bigsqcap\{A \in N_C \mid d \in A^{\mathcal{I}}\}$$
$$X^k(\mathcal{I},d) := X^0(\mathcal{I},d) \sqcap \bigsqcap_{r\in N_R} \bigsqcap\{\exists r.X^{k-1}(\mathcal{I},d') \mid (d,d') \in r^{\mathcal{I}}\}.$$

Chapter 3

Matching in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$

In this chapter, we define the matching problem for DLs $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$. In order to do that, we first introduce the notion of concept patterns and substitutions. Examples on how the solvability status of a specific instance of matching problem in $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ are affected by the concept constructors allowed and the types of TBoxes used will also be shown. We will also define special cases of matching problems whose computational complexity will be analysed in the subsequent chapters.

3.1 Concept patterns and substitutions

To define the $\mathcal{EL}$ matching problem, we partition the set of concept names N_C into two sets: one of concept variables N_v and one of concept constants N_c. A concept pattern is a concept that is constructed using $N_c \cup N_v$ as the set of concept names N_C using the constructors of $\mathcal{EL}$, without $\top$ in case of $\mathcal{EL}^{-\top}$. An $\mathcal{EL}$ concept description is ground if it does not contain any variables. A *substitution* σ is a finite mapping from concept variables N_v to concept constants N_c, i.e., there are only finitely many variables that are not mapped to itself. This mapping is extended to arbitrary concept descriptions as follows:

- $\sigma(A) := A$, for all $A \in N_C \cup \{\top\}$
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$
- $\sigma(\exists r.C) := \exists r.\sigma(C)$

Example 3.1.1. Let $N_v = \{X, Y\}$ and $N_c = \{A, B, C\}$. Consider a concept pattern,

$$C = X \sqcap A,$$

where X is a concept variable and $\{A, C\}$ are concept constants. Consider a substitution σ that maps X to B, i.e., $\sigma = \{X \mapsto B\}$. Applying σ to the concept pattern C, we get,

$$\sigma(C) = \sigma(X \sqcap A) = \sigma(X) \sqcap \sigma(A) = B \sqcap A.$$

In $\mathcal{EL}$, a substitution σ maps every concept variable to an $\mathcal{EL}$-concept description which may contain $\top$. However, in $\mathcal{EL}^{-\top}$, a substitution σ maps concept variables to $\mathcal{EL}^{-\top}$ concept descriptions. The substitution σ is called a *ground substitution* if for some variable X in N_v the concept description $\sigma(X)$ is ground. A general TBox $\mathcal{T}$ is called ground if no variable from N_v occurs in any GCI of $\mathcal{T}$.

3.2 Matching in $\mathcal{EL}$

An $\mathcal{EL}$ matching problem is defined as follows.

Definition 3.2.1. [BM14a] Let $\mathcal{T}$ be a ground general $\mathcal{EL}$-TBox. An $\mathcal{EL}$-matching problem w.r.t. $\mathcal{T}$ is a finite set $\Gamma := \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ of subsumption constraints between $\mathcal{EL}$-concept patterns, where for each $i, 1 \leq i \leq n$, C_i or D_i is ground. A substitution σ is an $\mathcal{EL}$ matcher of Γ w.r.t. $\mathcal{T}$ if σ solves all the subsumption constraints in Γ, i.e., if

$$\sigma(C_1) \sqsubseteq_{\mathcal{T}} \sigma(D_1), \ldots, \sigma(C_n) \sqsubseteq_{\mathcal{T}} \sigma(D_n).$$

We say that Γ is matchable w.r.t. $\mathcal{T}$ if it has a matcher.

An $\mathcal{EL}^{-\top}$ matching problem is defined in a similar way with one key difference that Γ is a finite set of subsumption constraints between $\mathcal{EL}^{-\top}$ concept patterns and a ground and general $\mathcal{EL}^{-\top}$-TBox is considered. In this case, the substitution that solves all subsumption constraints in Γ is now an $\mathcal{EL}^{-\top}$ matcher.

Matching problems modulo subsumption and equivalence are specific variants of the matching problem introduced in Definition 3.2.1 as also stated in [BM14a].

- The $\mathcal{EL}$ matching problem Γ is a matching problem modulo equivalence if $C \sqsubseteq^? D \in \Gamma$ implies $D \sqsubseteq^? C \in \Gamma$. We usually write $C \equiv D$ when referring to a matching problem modulo equivalence.
- The $\mathcal{EL}$ matching problem Γ is a left-ground matching problem modulo subsumption if $C \sqsubseteq^? D \in \Gamma$ implies that C is ground.
- The $\mathcal{EL}$ matching problem Γ is a right-ground matching problem modulo subsumption if $C \sqsubseteq^? D \in \Gamma$ implies that D is ground.

In this thesis, we investigate the computational complexity of the matching problems introduced above in $\mathcal{EL}^{-\top}$. We will consider both cases where $\mathcal{T} = \emptyset$ (i.e., no TBox is present) and when $\mathcal{T} \neq \emptyset$ (i.e., considering a ground and general $\mathcal{EL}^{-\top}$-TBox).

Example 3.2.2. Consider the following instance where $\mathcal{T}$ is empty,

$$\Gamma := \{A \sqsubseteq^? X,\ B \sqsubseteq^? X\},$$

where A and B are $\mathcal{EL}$ concepts and X is a variable. A substitution σ that replaces X by $\top$ is an $\mathcal{EL}$ matcher of Γ. However, Γ does not have an $\mathcal{EL}^{-\top}$ matcher.

Now, consider the case where a ground general $\mathcal{EL}$-TBox $\mathcal{T} := \{B \sqsubseteq A\}$ is present. The substitution σ satisfies the subsumptions $A \sqsubseteq_{\mathcal{T}} A$ and $B \sqsubseteq_{\mathcal{T}} A$, which implies that the concept name A is an $\mathcal{EL}^{-\top}$ matcher of Γ. Note that, the substitution $\sigma = \{X \mapsto \top\}$ is a matcher of Γ even in the presence of $\mathcal{T}$. Overall, Γ is a matching problem that not only has solutions with $\top$ but also without $\top$.

When the $\top$ concept is not allowed, the problem of finding a matcher becomes more difficult because $\top$ acts as a universal matcher. Without $\top$, matchers must be more specific. In the $\mathcal{EL}^{-\top}$ case, a substitution must find a concept that satisfies the subsumptions in Γ in a more constrained way. For example, the substitution $\sigma = \{X \mapsto \top\}$ trivially satisfies the conditions because both $A \sqsubseteq \top$ and $B \sqsubseteq \top$ hold. However, when $\top$ is not allowed, we need to find a more specific concept $\sigma(X)$ that simultaneously subsumes both A and B, i.e., $\sigma = \{X \mapsto A\}$ as $A \sqsubseteq_{\mathcal{T}} A$ and $B \sqsubseteq_{\mathcal{T}} A$ holds.

Finding a suitable $\mathcal{EL}^{-\top}$ matcher involves computing a specific concept that subsumes all the other concepts in Γ. This often requires analyz-

ing the structure of the concepts in the TBox (if one exists), making the problem more computationally demanding compared to when $\top$ is allowed. It involves more complex reasoning, particularly when dealing with large TBoxes or complex concept hierarchies. This makes the matching problem more challenging in $\mathcal{EL}^{-\top}$ compared to the standard $\mathcal{EL}$ setting.

3.3 Matching in $\mathcal{EL}^{-\top}$

In this section, we outline the decision problem considered in this thesis.

Given an instance of an $\mathcal{EL}^{-\top}$ matching problem Γ and a general $\mathcal{EL}^{-\top}$-TBox $\mathcal{T}$, as described in section 3.2, the goal is to determine whether Γ has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. $\mathcal{T}$.

In order to define such a problem, the notion of *normalized matching problems* is needed. This notion will be used in the algorithms for matching in $\mathcal{EL}^{-\top}$ presented in Chapter 5.

Definition 3.3.1. An $\mathcal{EL}$ matching problem is called normalized if $C \sqsubseteq^? D \in \Gamma$ implies that,

- either C or D is non-ground, and
- D is an atom.

Next, we provide some reductions between the different variants of the matching problem from Section 3.2. Based on this, in certain cases, we will be able to transfer lower bounds and upper bounds (complexity results) obtained for one variant into another.

Left-ground matching is a particular case of matching modulo equivalence. This is justified by the following lemma.

Lemma 3.3.2. *[BK99a] A substitution σ solves a left-ground $\mathcal{EL}^{-\top}$ matching problem $C \sqsubseteq^? D$ where C is ground iff it solves $C \equiv^? C \sqcap D$.*

From the lemma above, we can say that any left-ground matching problem can be transformed in polynomial time into an equivalent matching problem modulo equivalence, i.e., if a matcher for a left-ground matching problem modulo subsumption exists then it is also a matcher for the matching problem modulo equivalence. So, any lower bounds shown for left-ground matching also applies to matching modulo equivalence. An

algorithm solving matching problem modulo equivalence also solves a left-ground matching problem.

However, a matcher for a right-ground matching problem modulo subsumption is not a matcher for the matching problem modulo equivalence because a right-ground matching problem cannot be reduced to a matching problem modulo equivalence. Hence, the lower bound for right-ground matching problem cannot be applied to matching modulo equivalence.

Every matching problem modulo equivalence can be expressed as a general matching problem, since in both cases, variables can occur on either side of the subsumption constraints. Since left-ground matching is a particular case of matching modulo equivalence, and the latter a particular case of general matching, the lower-bounds are transferred in this direction, i.e., a lower bound for left-ground matching can be applied to matching modulo equivalence which in turn applies to general matching. Since all these problems are particular cases of general matching, an algorithm solving general matching also solves the other problems. Hence, an upper bound for general matching is also an upper bound for the the rest.

Chapter 4

Matching modulo subsumption and matching modulo equivalence in DL $\mathcal{EL}^{-\top}$

In this chapter, we will analyze the complexity of solving left-ground and right-ground matching modulo subsumption, and matching modulo equivalence for $\mathcal{EL}^{-\top}$. For this, we will consider the case of an empty TBox and the case where a general $\mathcal{EL}^{-\top}$-TBox is present.

For $\mathcal{EL}$, both left-ground and right-ground matching modulo subsumption have been shown to be in PTime in [BM14a], even in the presence of general $\mathcal{EL}$-TBoxes. In this thesis, we extend these results to $\mathcal{EL}^{-\top}$, where right-ground matching modulo subsumption remains in PTime, even when considering general $\mathcal{EL}^{-\top}$-TBoxes. However, for left-ground matching modulo subsumption, the complexity increases significantly in $\mathcal{EL}^{-\top}$. We show that it becomes NP-complete when dealing with an empty TBox and further increases to PSpace-hard in the presence of a general $\mathcal{EL}^{-\top}$-TBox. These results highlight a notable difference between $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ w.r.t. left-ground matching.

Next, we explore matching modulo equivalence in $\mathcal{EL}^{-\top}$. We show that the complexity remains in NP with an empty TBox, mirroring the result for $\mathcal{EL}$ as established in [BK99c]. Since left-ground matching is a particular case of matching modulo equivalence (as discussed in Section 3.3),

this also implies that both left-ground matching and matching modulo equivalence are NP-complete w.r.t. an empty TBox in $\mathcal{EL}^{-\top}$.

Finally, when considering general $\mathcal{EL}^{-\top}$-TBoxes, matching modulo equivalence becomes PSpace-hard, as inherited from the left-ground matching problem. While no specific algorithm is presented for left-ground matching or matching modulo equivalence in the presence of arbitrary TBoxes, the general matching algorithm presented in Chapter 5 can be applied, providing an upper bound of in ExpTime. Therefore, a gap remains in fully determining the complexity of left-ground matching and matching modulo equivalence w.r.t. general $\mathcal{EL}^{-\top}$-TBoxes. For simplicity, from now on we will refer to right-ground and left-ground matching problems modulo subsumption as right-ground and left-ground matching problems.

This chapter is structured as follows: we begin by examining the lower bound for left-ground matching in $\mathcal{EL}^{-\top}$, providing two key reductions. Following that, we present the complexity results for the upper bound of right-ground matching and matching modulo equivalence in $\mathcal{EL}^{-\top}$.

4.1 Lower bounds

4.1.1 NP-hardness of left-ground matching in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox

In this section, we show that left-ground matching in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox is NP-hard. To show this, we present a reduction from the well-known NP-complete problem - the **3SAT** problem.

A propositional formula in conjunctive normal form (CNF) is a conjunction of one or more clauses, where each clause is a disjunction of literals. Formally, a literal is either a propositional variable x or its negation $\neg x$. A CNF formula φ with n variables and m clauses can be expressed as,

$$\varphi = (l_{1,1} \vee l_{1,2} \vee \ldots \vee l_{1,k_1}) \wedge (l_{2,1} \vee l_{2,2} \vee \ldots \vee l_{2,k_2}) \wedge \ldots \wedge (l_{m,1} \vee l_{m,2} \vee \ldots \vee l_{m,k_m}),$$

where $l_{i,j}$ are literals, and each clause $(l_{i,1} \vee l_{i,2} \vee \ldots \vee l_{i,k_i})$ is a disjunction of literals for $1 \leq i \leq m$.

In the specific case of 3-CNF, each clause contains exactly three literals. Therefore, a 3-CNF formula is a CNF formula where each clause is restricted to three literals. An example of a 3-CNF formula is:

$$\varphi = (x_1 \vee \overline{x}_2 \vee x_3) \wedge (x_4 \vee \overline{x}_5 \vee \overline{x}_6) \wedge (\overline{x}_7 \vee \overline{x}_8 \vee x_9).$$

The 3-SAT problem asks whether there exists an assignment of truth values (*true* or *false*) to the variables in a 3-CNF formula that makes the entire formula true. If such an assignment exists, the formula is satisfiable; otherwise, it is unsatisfiable. The 3-SAT problem is known to be NP-complete [GJ79].

We adapt the idea from [BK00], where it was shown that the complexity of solving matching modulo equivalence in $\mathcal{EL}$ w.r.t. an empty TBox is NP-hard. For the reduction, we consider a propositional formula φ in 3CNF, where each clause only contains three literals. For any such propositional formula φ, we construct a left-ground $\mathcal{EL}^{-\top}$ matching problem Γ_φ. This construction encodes the satisfiability of φ into a matching problem, where finding an $\mathcal{EL}^{-\top}$ matcher corresponds to finding a satisfying assignment for the variables of φ. The encoding is done in a way such that a solution to the matching problem exists iff the original propositional formula is satisfiable, i.e.,

$$\varphi \text{ is satisfiable } \textbf{iff } \Gamma_\varphi \text{ has an } \mathcal{EL}^{-\top} \text{ matcher.}$$

The reduction for the construction of Γ_φ is defined in two steps:

Step 1: Propositional assignment encoding:

We define a propositional assignment encoding where each literal x_i and $\neg x_i$ in φ is associated to concept variables X_i and $\overline{X}_i$, with $1 \leq i \leq n$, and n stands for the number of propositional variables in φ. In addition, we use concept constants A_t and A_f to represent *true* and *false*.

Let r be role name and i an index with $1 \leq i \leq n$. We define the following subsumption constraints:

$$\begin{aligned} C := \exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \\ \sqsubseteq^? \\ D := \exists r.(X_1 \sqcap \exists r.\overline{X}_1) \sqcap \ldots \sqcap \exists r.(X_n \sqcap \exists r.\overline{X}_n) \end{aligned} \tag{4.1}$$

and,

$$\begin{array}{c} P := A_t \sqcap A_f \\ \sqsubseteq^{?} \\ Q := X_1 \sqcap \overline{X}_1 \sqcap \ldots \sqcap X_n \sqcap \overline{X}_n. \end{array} \tag{4.2}$$

Lemma 4.1.1. *For all $\mathcal{EL}^{-\top}$ substitutions σ,*

$$\sigma(C) \sqsubseteq \sigma(D) \text{ and } P \sqsubseteq Q$$
$$\textit{iff}$$
$$(\sigma(X_i) \equiv A_t \wedge \sigma(\overline{X}_i) \equiv A_f) \text{ or } (\sigma(X_i) \equiv A_f \wedge \sigma(\overline{X}_i) \equiv A_t),$$

for every i, where $1 \leq i \leq n$.

Proof. ($\Rightarrow$) Assuming that a substitution σ solves the subsumption relations $\sigma(C) \sqsubseteq \sigma(D)$ and $P \sqsubseteq Q$, we show that the properties on the right-hand side (R.H.S.) of the lemma are also satisfied. To show the implication, it is enough to prove it for an arbitrary i. Hence, the following subsumption relations also hold:

$$\begin{array}{c} \exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \\ \sqsubseteq \\ \exists r.(X_1 \sqcap \exists r.\overline{X}_1) \end{array} \tag{4.3}$$

and,

$$\begin{array}{c} A_t \sqcap A_f \\ \sqsubseteq \\ X_1 \sqcap \overline{X}_1 \end{array} \tag{4.4}$$

From the subsumption $P \sqsubseteq Q$, we know that X_1 and $\overline{X}_1$ can only be substituted with concept constants A_t (representing *"true"*) and A_f (representing *"false"*). This is because σ cannot assign $\top$ to the variables as it a substitution in $\mathcal{EL}^{-\top}$. In particular, the structure $P := A_t \sqcap A_f$ on the L.H.S. ensures that X_1 and $\overline{X}_1$ must be assigned concept constants A_t and A_f or a conjunction of both.

It remains to show that X_1 and $\overline{X}_1$ gets assigned different concept constants. Considering ($X_1 \equiv A_t$ and $\overline{X}_1 \equiv A_t$) or ($X_1 \equiv A_f$ and $\overline{X}_1 \equiv A_f$), we prove by contradiction that the subsumption $\sigma(C) \sqsubseteq \sigma(D)$ will no longer hold as it contradicts our initial assumption on σ. The

subsumption relation 4.3 results in,

$$\exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \sqsubseteq \exists r.(A_t \sqcap \exists r.A_t)$$

or

$$\exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \sqsubseteq \exists r.(A_f \sqcap \exists r.A_f).$$

The above subsumption does not hold because the concept C is no longer subsumed by the concept D as the subsumption relation is violated. This can be said using the characterization of subsumption w.r.t. empty TBoxes shown in Lemma 2.2.1. Since the variables cannot be substituted with the same concept constants or with existential restrictions (from subsumption $P \sqsubseteq Q$), we conclude that the substitution σ must replace each variable with either A_t or A_f. Specifically, $(\sigma(X_i) \equiv A_t \wedge \sigma(\overline{X}_i) \equiv A_f)$ or $(\sigma(X_i) \equiv A_f \wedge \sigma(\overline{X}_i) \equiv A_t)$.

($\Leftarrow$) Assuming that a substitution σ satisfies the properties on the R.H.S. of Lemma 4.1.1, i.e., $(\sigma(X_i) \equiv A_t \wedge \sigma(\overline{X}_i) \equiv A_f)$ or $(\sigma(X_i) \equiv A_f \wedge \sigma(\overline{X}_i) \equiv A_t)$, we prove that σ also satisfies $\sigma(C) \sqsubseteq \sigma(D)$ and $P \sqsubseteq Q$.

It is enough to show that $C \sqsubseteq \sigma(\exists r.(X_i \sqcap \exists r.\overline{X}_i))$ and $P \sqsubseteq \sigma(X_i \sqcap \overline{X}_i)$ holds for all i, i.e.,

$$\exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \sqsubseteq \exists r.(A_t \sqcap \exists r.A_f)$$

or

$$\exists r.(A_t \sqcap \exists r.A_f) \sqcap \exists r.(A_f \sqcap \exists r.A_t) \sqsubseteq \exists r.(A_f \sqcap \exists r.A_t)$$

and,

$$A_t \sqcap A_f \sqsubseteq A_t \sqcap A_f$$

or

$$A_t \sqcap A_f \sqsubseteq A_f \sqcap A_t.$$

The substitution σ clearly does not violate the subsumption relations above as the concept on the right-hand side subsumes the concept on the left-hand side. $\sigma(C) \sqsubseteq \sigma(D)$ and $P \sqsubseteq Q$ hold upto associativity and commutativity of conjunction. Therefore, the properties on the R.H.S. of the proposition do not conflict with the subsumptions on the L.H.S. of the Lemma 4.1.1. □

Step 2: Simulating the satisfiability of φ

To simulate the satisfiability of a propositional formula φ, we associate each clause c_j in φ (with $1 \leq j \leq m$, where m denotes the number of clauses) with a concept pattern H_j as follows,

$$c_j = l_{j1} \vee l_{j2} \vee l_{j3} \longrightarrow H_j := Z_{j1} \sqcap Z_{j2} \sqcap Z_{j3}, \tag{4.5}$$

where $Z_{jh} = X_i$ if $l_{jh} = x_i$ and $Z_{jh} = \overline{X}_i$ if $l_{jh} = \neg x_i$ and $h \in \{1, 2, 3\}$. Each literal l_{jh} corresponds to a concept pattern Z_{jh}.

We define the Equation 4.5 for one clause, where s is role name different from r that **does not** occur in concepts C or D in the subsumption constraint in 4.1 as follows,

$$\begin{gathered} G := \exists s.(A_t \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap A_f))) \\ \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap \exists s.(A_t \sqcap A_f))) \\ \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.A_t)) \\ \sqsubseteq^? \\ H_j := \exists s.(Z_{j1} \sqcap \exists s.(Z_{j2} \sqcap \exists s.Z_{j3})). \end{gathered} \tag{4.6}$$

The same definition can be extended to other clauses in φ.

Lemma 4.1.2. *For all $\mathcal{EL}^{-\top}$ substitutions σ,*

$$\sigma(G) \sqsubseteq \sigma(H_j) \text{ and } P \sqsubseteq Q$$

iff

$$\sigma(Z_{jh}) \equiv A_t, \text{ for at least one } h \in \{1, 2, 3\},$$

for every j, where $1 \leq j \leq m$ and m denotes the number of clauses in φ.

Proof. ($\Rightarrow$) Consider a substitution σ that satisfies the subsumptions $\sigma(G) \sqsubseteq \sigma(H_j)$ and $P \sqsubseteq Q$. We prove that σ also satisfies the property, that at least one of the concept variables Z_{jh} is replaced with A_t, i.e., $\sigma(Z_{jh}) \equiv A_t$ for some $h \in \{1, 2, 3\}$. We prove this by contradiction.

From the subsumption $P \sqsubseteq Q$, it follows that the variables X_1 and $\overline{X}_1$ can only be replaced by the concept constants A_t (representing *"true"*) and A_f (representing *"false"*). This is due to the fact that, in $\mathcal{EL}^{-\top}$, the substitution σ is restricted and cannot map variables to $\top$, since $\top$ is not part of the concept constructors.

Applying the substitution σ to 4.6 yields,

$$\exists s.(A_t \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap A_f))) \sqsubseteq \exists s.(A_f \sqcap \exists s.(A_f \sqcap \exists s.A_f))$$

or

$$\exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap \exists s.(A_t \sqcap A_f))) \sqsubseteq \exists s.(A_f \sqcap \exists s.(A_f \sqcap \exists s.A_f))$$

or

$$\exists s.((A_t \sqcap A_f) \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.A_t)) \sqsubseteq \exists s.(A_f \sqcap \exists s.(A_f \sqcap \exists s.A_f))$$

In this case, if none of the concept variables Z_{j1}, Z_{j2} or Z_{j3} are substituted with A_t, then all variables are substituted with A_f, which would mean that the left-hand side no longer subsumes the right-hand side. Therefore, at least one of the Z_{jh} must be substituted with A_t, ensuring that at least one concept variable matches the pattern. We assume no Z_{jh} to be replaced by A_t, because for the formula φ to be satisfiable, at least one of the conjunct should be true, which implies for the reduction to $\mathcal{EL}^{-\top}$, at least one of the concept variables Z_{jh} must be substituted with A_t.

The concept on the R.H.S. of the subsumption above no longer subsumes the concept on the L.H.S., which implies that the L.H.S. of the lemma 4.1.2 is true only when at least one of Z_{jh} is replaced by A_t.

($\Leftarrow$) For the if direction, we show that every substitution σ that replaces at least one of the concept variables Z_{jh} with A_t, also satisfies $\sigma(G) \sqsubseteq \sigma(H_j)$ and $P \sqsubseteq Q$. Applying σ to the equation 4.6 and replacing at least one of concept variables Z_{j1} or Z_{j2} or Z_{j3} with A_t results in the following,

$$\exists s.(A_t \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap A_f))) \sqsubseteq \exists s.(A_t \sqcap \exists s.(A_f \sqcap \exists s.A_f))$$

or

$$\exists s.((A_t \sqcap A_f) \sqcap \exists s.(A_t \sqcap \exists s.(A_t \sqcap A_f))) \sqsubseteq \exists s.(A_f \sqcap \exists s.(A_t \sqcap \exists s.A_f))$$

or

$$\exists s.((A_t \sqcap A_f) \sqcap \exists s.((A_t \sqcap A_f) \sqcap \exists s.A_t)) \sqsubseteq \exists s.(A_f \sqcap \exists s.(A_f \sqcap \exists s.A_t))$$

The subsumption constraint 4.2 results in,

$$A_t \sqcap A_f \sqsubseteq A_t \sqcap A_f$$

This follows from 4.5 and Lemma 4.1.1, where $Z_{jh} \equiv X_i$ implies $X_i \equiv A_t$ and $Z_{jh} \equiv \overline{X}_i$ implies $X_i \equiv A_f$. The subsumption relation for the above clearly holds which implies that when at least one of Z_{jh} is replaced with A_t, the L.H.S. of Lemma 4.1.2 also holds. □

Next, we will see how we can combine the previous two steps to construct from φ, a left-ground $\mathcal{EL}^{-\top}$ matching problem Γ_φ. The final construction is as follows:

Let $s_1, \ldots, s_m$ be distinct role names for each clause c_j not occurring in C nor in D. Then,

$$\begin{aligned} \Gamma_\varphi := \{ & C \sqsubseteq^? D,\ \exists s_1.G \sqcap \ldots \sqcap \exists s_m.G \sqsubseteq^? \\ & \exists s_1.H_1 \sqcap \ldots \sqcap \exists s_m.H_m,\ P \sqsubseteq^? Q\} \end{aligned} \tag{4.7}$$

Lemma 4.1.3. *φ is satisfiable iff Γ_φ has a matcher in $\mathcal{EL}^{-\top}$.*

Proof. ($\Rightarrow$) Suppose φ is a satisfiable propositional formula, then there exists a truth assignment t that satisfies φ. We build a substitution σ_t from this truth assignment. If the propositional variable x_i is *true*, then A_t gets assigned to the concept variable X_i and A_f to $\overline{X}_i$, and if x_i is *false*, then A_f is assigned to the concept variable X_i and A_t is assigned to $\overline{X}_i$. This is shown below,

$$\begin{aligned} \sigma_t(X_i) = A_t \text{ and } \sigma_t(\overline{X}_i) = A_f; \qquad & \text{when } t(x_i) = true \\ \sigma_t(X_i) = A_f \text{ and } \sigma_t(\overline{X}_i) = A_t; \qquad & \text{when } t(x_i) = false \end{aligned}$$

By the construction of σ_t, we can say that the substitution matches the property $(\sigma(X_i) \equiv A_t \wedge \sigma(\overline{X}_i) \equiv A_f)$ or $(\sigma(X_i) \equiv A_f \wedge \sigma(\overline{X}_i) \equiv A_t)$ from Lemma 4.1.1. Applying the lemma, we obtain that the subsumptions $\sigma_t(C) \sqsubseteq \sigma_t(D)$ and $\sigma_t(P) \sqsubseteq \sigma_t(Q)$ holds.

It remains to show that the subsumption constraint $\exists s_1.G \sqcap \ldots \sqcap \exists s_m.G \sqsubseteq^? \exists s_1.H_1 \sqcap \ldots \sqcap \exists s_m.H_m$ also holds. For the first part, since we considered a different role name for each of the clauses in φ, we know that $s_i \neq s_j$, for all $1 \leq i, j \leq m$. This is the same as showing $\sigma_t(G) \sqsubseteq \sigma_t(H_i)$ for all $1 \leq i \leq m$. Consider an arbitrary i, with $1 \leq i \leq m$. Since t satisfies the propositional formula φ or $t \models \varphi$, we know there exists a literal c_{jh} in the clause c_j such that $t \models c_{jh}$, because to satisfy a clause at least one of the literals must be satisfied. We will now consider both instances where the literal is (a) a positive instance, $x_i = true$ and (b) a negative instance, $x_i = false$.

- If $c_{jh} = x_i$, then $t(x_i) = true$, this means $\sigma_t(X_i) = A_t$ by construction of the substitution σ_t. Hence, $\sigma_t(Z_{jh}) = A_t$, because $Z_{jh} = X_i$ (from 4.5) by the definition of the concept pattern H_j. Therefore, from Lemma 4.1.2 we obtain that $\sigma_t(G) \sqsubseteq \sigma_t(H)$ or $\sigma(G) \sqsubseteq \sigma(H)$ holds.

- If $c_{jh} = \neg x_i$, then $t(x_i) = false$, this means $\sigma_t(\overline{X}_i) = A_t$ by construction. Hence, $\sigma_t(Z_{jh}) = A_t$, because $Z_{jh} = \overline{X}_i$. From Lemma 4.1.2, we have $\sigma_t(G) \sqsubseteq \sigma_t(H)$ or $\sigma(G) \sqsubseteq \sigma(H)$.

Hence, we can say that the substitution σ_t that we built from the propositional assignment t satisfies all the subsumptions of our matching problem Γ_φ. This proves the only-if direction.

($\Leftarrow$) For this direction, we assume that the matching problem Γ_φ has a matcher σ_t that solves all the subsumptions in Γ_φ. We use this substitution σ_t to build a truth assignment that satisfies the propositional formula φ.

We have shown that every clause in φ can be represented as a concept pattern H_j as shown in 4.5. From 4.1.2, we know that to satisfy a clause at least one of the literals must be mapped to a positive instance, i.e., $x_i = true$ or $X_i = A_t$. From 4.5 we can see that, $Z_{jh} = A_t$ when $l_{jh} = x_i$ and $x_i = true$ or vice-versa. By Lemma 4.1.1, we can see that when σ_t is matcher of $\sigma_t(C) \sqsubseteq \sigma_t(D)$, then the variables X_i and

$\overline{X}_i$ are always mapped to A_t and A_f respectively or vice-versa. The subsumption $\sigma_t(P) \sqsubseteq \sigma_t(Q)$ ensures that the variables are only mapped to concept constants A_t and A_f for *true* and *false* and not existential restrictions or others.

To sum up, the properties used by the substitution σ_t, simulates the propositional assignment encodings and satisfiability of clauses in the propositional formula φ. Hence, we can say that whenever Γ_φ has an $\mathcal{EL}^{-\top}$ matcher σ_t, then φ has a satisfying truth assignment t. □

Reduction from 3SAT to left-ground matching in $\mathcal{EL}^{-\top}$ takes polynomial time

To show that this reduction takes PTime, we take any propositional formula φ and construct a left-ground matching problem Γ_φ in $\mathcal{EL}^{-\top}$. This construction can be done in polynomial time in the size of the propositional formula because the concept patterns that we build in Γ_φ are of polynomial size and there is no operation involved of more than polynomial time to construct these concept patterns from φ. Since Lemma 4.1.3 shows that our reduction is correct, we obtain the following lower bound for left-ground matching in $\mathcal{EL}^{-\top}$.

Theorem 4.1.4. *Deciding whether a left-ground $\mathcal{EL}^{-\top}$-matching problem modulo subsumption has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. an empty TBox is NP-hard.*

4.1.2 PSpace-hardness of left-ground matching in $\mathcal{EL}^{-\top}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox

In this section, we will show that left-ground matching modulo subsumption in $\mathcal{EL}^{-\top}$ in the presence of a general $\mathcal{EL}^{-\top}$-TBox is PSpace-hard. We prove this by reducing the intersection emptiness problem for a sequence of deterministic finite automatas (DFAs), which is known to be PSpace-hard [Koz77], to left-ground matching in $\mathcal{EL}^{-\top}$ using TBox axioms. Let us first understand the idea behind this reduction and later see how can we adapt it to matching in $\mathcal{EL}^{-\top}$.

Definition 4.1.5. A *deterministic finite automaton* (DFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ consisting of,

- a finite set of *states* Q,

- a finite alphabet Σ of *input symbols*,
- an *initial state* $q_0 \in Q$,
- a *transition function* $\delta : Q \times \Sigma \to Q$ and
- a set $F \subseteq Q$ of *final states*.

A *configuration* of $\mathcal{A}$ is a pair (q, w), where $q \in Q$ and $w \in \Sigma^*$. The transition function δ induces the following binary relation $\vdash_{\mathcal{A}}$ between configurations: $(q, w) \vdash_{\mathcal{A}} (q', w')$ iff either

- $w = w'$ and $q' \in \delta(q, \varepsilon)$ (ε-transition) or
- $w = \alpha w'$ and $q' \in \delta(q, \alpha)$ for some $\alpha \in \Sigma$ (α-transition).

The second kind of transition is only possible if $w \neq \varepsilon$, i.e., there is still a part of the input word left to read.

A *run* of $\mathcal{A}$ is a finite, nonempty tree labeled by configurations of $\mathcal{A}$. An input word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ iff there is a successful run of $\mathcal{A}$, the root of which is labeled by (q_0, w). The language recognized by $\mathcal{A}$ is $L(\mathcal{A}) := \{w \in \Sigma_* \mid w \text{ is accepted by } \mathcal{A}\}$.

The intersection emptiness problem considers finitely many DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$, and asks whether $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_k) \neq \emptyset$. Since this problem is trivially solvable in polynomial time in case $L(\mathcal{A}_i) = \emptyset$ for some $1 \leq i \leq k$, all languages $L(\mathcal{A}_i)$ are assumed to be nonempty.

In [Baa+11a], each DFA $\mathcal{A}_i$ (where $1 \leq i \leq n$) in the sequence of automatas is translated into a set of subsumption constraints $\Gamma_{\mathcal{A}_i}$. The set of constraints $\Gamma_{\mathcal{A}_i}$ is meant to simulate the behaviour of $\mathcal{A}$. Variables are used on both sides (unification) to capture loops in the automaton. It can also be assumed without loss of generality, that the automata $\mathcal{A}_i = (Q_i, \Sigma, q_{0,i}, \delta_i, F_i)$ have pairwise disjoint sets of states Q_i, i.e., there is no state that cannot be reached from the initial state or from which no final state can be reached. Such states can be removed from $\mathcal{A}$ without changing the accepted language.

The subsumption constraints in the unification problem are considered to be flat, i.e., they consist of equations between flat concept terms, defined in 2.1.1. By introducing new concept variables and eliminating $\top$, every $\mathcal{EL}^{-\top}$-unification problem Γ can be transformed in polynomial time into a flat $\mathcal{EL}^{-\top}$-unification problem Γ', such that Γ is solvable iff

Γ' is solvable [BM10].

To capture the set of words accepted by the automaton $\mathcal{A}$, the property that "if σ is a ground $\mathcal{EL}^{-\top}$ unifier of $\Gamma_{\mathcal{A}}$ with $\sigma(X_q) \sqsubseteq \exists w.A$ then $w \in L(\mathcal{A}_q)$ where q is the initial state" was used. Conversely, "if $w \in L(\mathcal{A}_q)$ with q as the initial state then σ is an $\mathcal{EL}^{-\top}$ unifier of $\Gamma_{\mathcal{A}}$ with $\sigma(X_q) \sqsubseteq \exists w.A$". This property becomes important when dealing with the intersection emptiness problem, where we are concerned with finding whether a common word that is accepted by a sequence of automatas $\mathcal{A}_i$ for $1 \leq i \leq k$. To address this, a flat $\mathcal{EL}^{-\top}$ unification problem was formulated by combining all the constraints $\Gamma_{\mathcal{A}_i}$ (from each automaton $\mathcal{A}_i$) in a way that captures a common word accepted by all automatas. The unification problem is $\mathcal{EL}^{-\top}$-unifiable iff the intersection of the languages of the automata is non-empty, i.e., $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_k) \neq \emptyset$.

Keeping in mind the assumptions previously discussed and the properties above, we will adopt an approach similar to the one outlined in [Baa+11a], but with a key difference - we allow variables on only one side of the subsumption constraints. It is necessary because we want to reduce the intersection emptiness problem to matching. Additionally, the TBox is used to simulate the behavior of automatas $\mathcal{A}_i$ that accepts a common word. This adaptation preserves the core idea while addressing the specific requirements of matching in $\mathcal{EL}^{-\top}$.

Given an automata $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. For every state $q \in Q$, we introduce a new concept constant B_q. Let $\mathcal{A}_q = (Q, \Sigma, q_0, \delta, F)$ be an automata obtained from $\mathcal{A}$, by making q the initial state. With $N_R = \Sigma$, the set of GCIs in the TBox $\mathcal{T}_{\mathcal{A}}$ are defined as follows,

$$\begin{aligned}\mathcal{T}_{\mathcal{A}} := \{B_{q_f} \sqsubseteq A \mid q \in F\} \cup \\ \{B_q \sqsubseteq \prod_{\substack{\alpha \in \Sigma \\ \delta(q,\alpha) \text{ is defined}}} \exists \alpha.B_{\delta(q,\alpha)} \mid q \in Q \setminus F\},\end{aligned}$$

where B_{q_f} and A are concept constants.

The first GCI represents the final state where there is no input symbol left to read and the second GCI represents the transition from one state to another $\delta : q \times \alpha \to q'$, with an input symbol α.

We prove the following,

Lemma 4.1.6. *Let $q \in Q$, $w \in \Sigma^*$. If $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w.A$, then $w \in L(\mathcal{A}_q)$.*

Proof. We prove this by induction on the length of w.

For the base case, if $|w| = 0$, then q must be a final state in $w \in L(\mathcal{A}_q)$. This implies $\exists w.A = A$, which means the empty word is accepted by $\mathcal{A}_q$. Since, $|w| = 0$ implies $q \in F$, then $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} A$.

For the inductive step, let $w = \alpha' w'$ with $\alpha' \in \Sigma$, $w' \in \Sigma^*$. Since $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w.A$ then from the second GCI in $\mathcal{T}$, it follows that $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists \alpha'.\exists w'.A$. By structure of subsumption in Lemma 2.2.3, this implies that there exists a state q' such that, $B_{\delta(q,\alpha')} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w'.A$. By induction, we know that w' is accepted by $\mathcal{A}_{\delta(q,\alpha')}$. Thus, $w = \alpha' w'$ is also accepted by $\mathcal{A}_q$. □

From the proof above we can assume that the word corresponding to the language accepted by the automata $\mathcal{A}_q$ is a particle that is always subsumed by $\exists w.A$.

Note that, from every state $q \in Q$ there always exists a word u_q of minimum length that is accepted by $\mathcal{A}_q$ as we have assumed that we can reach F from every state.

Lemma 4.1.7. *If $w \in L(\mathcal{A}_q)$, then $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w.A$.*

Proof. Let the unique successful run of $\mathcal{A}$ on $w = w_1 \dots w_n$ be given by the sequence $q_0 q_1 \dots q_n$ of states with $q_n \in F$ and $\delta(q_i, w_{i+1}) = q_{i+1}$ for every $i \in \{0, \dots, n-1\}$. We need to show that $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w.A$ holds. In other words, for each state q_i along the run, the concept B_{q_i} is subsumed by $\exists w_{i+1} \dots w_n.A$.

1. Consider $\exists u_q.A$.
   - If $u_q = \varepsilon$ (the empty word), then $q \in F$. In this case, $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} A$, so the subsumption $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists u_q.A$ is trivially satisfied.
   - Otherwise, by construction there is a transition $\delta(q, \alpha) = q'$ with $u_q = \alpha u_{q'}$. Since, $\exists u_{q'}.A$ subsumes $B_{q'}$, we have $B_{q'} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists u_{q'}.A$. Therefore, $\exists \alpha.B_{q'} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists u_{q'}.A$, which implies that $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists u_q.A$.
2. Consider $\exists w_{i+1} \dots w_n.A$ where $q_i \in \{q_0, \dots, q_{n-1}\}$.

- Since, $\delta(q_i, w_{i+1}) = q_{i+1}$ and by induction $B_{q_{i+1}} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w_{i+2} \dots w_n.A$ we know that $\exists w_{i+1}.B_{q_{i+1}} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w_{i+1} \dots w_n.A$. Thus, B_{q_i} is subsumed by $\exists w_{i+1} \dots w_n.A$, i.e., $B_{q_i} \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w_{i+1} \dots w_n.A$

This shows that for each state q_i in the successful run of $\mathcal{A}_q$ on w, the concept B_{q_i} is subsumed by $\exists w_{i+1} \dots w_n.A$. Specifically, since the run starts at state q_0 and ends in an accepting state q_n, we have $B_q \sqsubseteq_{\mathcal{T}_\mathcal{A}} \exists w_1 \dots w_n.A = \exists w.A$. This completes the proof of the reduction in both directions. □

The intersection emptiness problem considers finitely many automatas $\mathcal{A}_1, \dots, \mathcal{A}_k$ with $1 \leq k \leq n$ and asks whether $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k) \neq \emptyset$. Therefore, we need to consider one TBox $\mathcal{T}_\mathcal{A}$ for every automata $\mathcal{A}_k$.

The general $\mathcal{EL}^{-\top}$-TBox is defined as follows,

$$\mathcal{T} := \bigcup_{i \in \{1,\dots,k\}} \mathcal{T}_{\mathcal{A}_i}$$

and the left-ground $\mathcal{EL}^{-\top}$ matching problem is defined as follows,

$$\Gamma := \{ \bigcup_{i \in \{1,\dots,k\}} B_{q_{0,i}} \sqsubseteq^? Y \},$$

where Y is a new variable.

Lemma 4.1.8. *Γ has an $\mathcal{EL}^{-\top}$ matcher w.r.t. $\mathcal{T}$ iff $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k) \neq \emptyset$.*

Proof. ($\Rightarrow$) Assume that Γ has an $\mathcal{EL}^{-\top}$ matcher σ. Then, the substitution σ satisfies the subsumption $B_{q_{0,i}} \sqsubseteq_\mathcal{T} \sigma(Y)$ for all $i \in \{1, \dots, k\}$. We know that, $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}_{\mathcal{A},i}} \sigma(Y)$ and $\sigma(Y) \sqsubseteq_\mathcal{T} \exists w.A$ holds for some word $w \in \Sigma^*$. This is because except for the concept name A the TBoxes $\mathcal{T}_{\mathcal{A}_i}$ used to build $\mathcal{T}$ do not share other concept names. Transitivity of subsumption implies that $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}_{\mathcal{A},i}} \exists w.A$. Applying Lemma 4.1.6 to $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}_{\mathcal{A},i}} \exists w.A$, it follows that $w \in L(\mathcal{A}_i)$ for every i. Therefore, $w \in L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k)$, implying that the intersection of the languages $L(\mathcal{A}_i)$ is nonempty.

($\Leftarrow$) Suppose there exists a word $w \in \Sigma^*$ such that $w \in L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k)$. By lemma 4.1.7, since $w \in L(\mathcal{A}_i)$ for each $i \in \{1, \dots, k\}$, we have $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}_{\mathcal{A},i}} \exists w.A$ for every i. We define a substitution σ as

$\sigma(Y) := \exists w.A$ and we have $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}_{\mathcal{A},i}} \exists w.A$ for each q in the respective states of $\mathcal{A}_i$. Given that $\mathcal{T} := \bigcup_i \mathcal{T}_{\mathcal{A}_i}$ and considering Γ, the substitution $B_{q_{0,i}} \sqsubseteq_{\mathcal{T}} \sigma(Y)$ holds for all i. Therefore, σ is an $\mathcal{EL}^{-\top}$ matcher of Γ. □

Theorem 4.1.9. *Deciding whether a left-ground $\mathcal{EL}^{-\top}$-matching problem modulo subsumption has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. a general $\mathcal{EL}^{-\top}$-TBox is PSpace-hard.*

For the complexity of matching modulo equivalence for $\mathcal{EL}^{-\top}$ in the presence of a general $\mathcal{EL}^{-\top}$-TBox, we have shown in theorem above that left-ground $\mathcal{EL}^{-\top}$ matching modulo subsumption w.r.t. a general $\mathcal{EL}^{-\top}$-TBox is PSpace-hard. As we have reduced the intersection emptiness problem for a sequence of DFAs (general case in unification) to left-ground $\mathcal{EL}^{-\top}$ matching w.r.t. a general $\mathcal{EL}^{-\top}$-TBox (particular case), we can say that this result also holds for matching modulo equivalence in $\mathcal{EL}^{-\top}$ as stated in the theorem below.

Theorem 4.1.10. *Deciding whether a matching problem modulo equivalence in $\mathcal{EL}^{-\top}$ has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. a general $\mathcal{EL}^{-\top}$-TBox is PSpace-hard.*

4.2 Upper bounds

In this section, we will show that the complexity of the right-ground matching problem in $\mathcal{EL}^{-\top}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox remains in PTime. We will also show that matching modulo equivalence w.r.t. the empty TBox in $\mathcal{EL}^{-\top}$ is in NP. This, together with the NP-hard lower bound shown for left-ground matching in Theorem 4.2.7, implies that the problem is NP-complete, the same as for EL.

4.2.1 PTime complexity of right-ground matching in $\mathcal{EL}^{-\top}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox

For $\mathcal{EL}$, deciding whether a right-ground matching problem has a matcher or not w.r.t. a general $\mathcal{EL}$-TBox can be done in PTime [BM14a]. We will show that this is also the case for $\mathcal{EL}^{-\top}$.

Given a general $\mathcal{EL}$-TBox $\mathcal{T}$ and a right-ground matching problem $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$. The $\mathcal{EL}$ concept description $\bot\,(\Gamma, \mathcal{T})$ is used to denote the $\mathcal{EL}$-concept description that is the conjunction

of all the atoms of $\mathcal{T}$ and of $D_1, \ldots, D_n$. The substitution $\sigma_{\perp(\Gamma,\mathcal{T})}$ is defined as $\sigma_{\perp(\Gamma,\mathcal{T})}(X) = \perp (\Gamma, \mathcal{T})$ for all $X \in N_V$.

Lemma 4.2.1. *[BM14a] Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a right-ground $\mathcal{EL}$ matching problem modulo subsumption and $\mathcal{T}$ be a general $\mathcal{EL}$-TBox. Then Γ has a matcher w.r.t. $\mathcal{T}$ iff $\sigma_{\perp(\Gamma,\mathcal{T})}$ is a matcher of Γ w.r.t. $\mathcal{T}$.*

We can restrict our attention to $\sigma_{\perp(\Gamma,\mathcal{T})}$ as it is the *most specific concept* that can be considered as a matcher of the problem. By the construction of $\sigma_{\perp(\Gamma,\mathcal{T})}$, it is easy to see that $\sigma_{\perp(\Gamma,\mathcal{T})}$ is also an $\mathcal{EL}^{-\top}$ substitution, if the problem being considered was an $\mathcal{EL}^{-\top}$ right-ground matching problem.

Example 4.2.2. Let us understand this with the help of an example. Consider a right-ground matching problem Γ in $\mathcal{EL}^{-\top}$, where the concept $\sigma_{\perp(\Gamma,\mathcal{T})}$ does not contain $\top$. Consider,

$$\mathcal{T} = \{\exists s.B \sqsubseteq \exists s.C,\ C \sqsubseteq \exists s.C,\ C \sqsubseteq A\}$$
$$\Gamma = \{X \sqcap B \sqsubseteq^? \exists s.A\}$$

The matcher of Γ is defined as $\sigma_{\perp(\Gamma,\mathcal{T})}(X) = \perp (\Gamma, \mathcal{T})$ for all $X \in N_V$. $\perp (\Gamma, \mathcal{T})$ represents the conjunction of all atoms in $\mathcal{T}$ and of $D_1, \ldots, D_n$, i.e.,

$$\sigma_{\perp(\Gamma,\mathcal{T})}(X) = A \sqcap B \sqcap C \sqcap \exists s.A \sqcap \exists s.B \sqcap \exists s.C$$

It is evident that $\sigma_{\perp(\Gamma,\mathcal{T})}(X)$ does not contain $\top$ due to its construction. It is easy to see that the subsumption $X \sqcap B \sqsubseteq^? \exists s.A$ of Γ, the subsumption $\sigma_{\perp(\Gamma,\mathcal{T})}(X) \sqcap B \sqsubseteq_{\mathcal{T}} \exists s.A$ holds. By construction of $\sigma_{\perp(\Gamma,\mathcal{T})}$ it is clear that it is an $\mathcal{EL}^{-\top}$-concept description, which also makes it an $\mathcal{EL}^{-\top}$-matcher of the problem.

Lemma 4.2.3. *Let $\Gamma = \{C_1 \sqsubseteq^? D_1, \ldots, C_n \sqsubseteq^? D_n\}$ be a right-ground $\mathcal{EL}^{-\top}$ matching problem modulo subsumption and $\mathcal{T}$ be a general $\mathcal{EL}^{-\top}$-TBox. Then Γ has an $\mathcal{EL}^{-\top}$-matcher w.r.t. $\mathcal{T}$ iff $\sigma_{\perp(\Gamma,\mathcal{T})}$ is an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. The "if" direction is trivial. For the "only-if" direction, assume that σ is an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$. We need to show that this implies $\sigma_{\perp(\Gamma,\mathcal{T})}$ is also an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$, i.e., it satisfies

$\sigma_{\perp(\Gamma,\mathcal{T})}(C) \sqsubseteq_{\mathcal{T}} \sigma_{\perp(\Gamma,\mathcal{T})}(D)$ for every subsumption $C \sqsubseteq^? D \in \Gamma$. Since, σ is an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$, for every subsumption $C_i \sqsubseteq^? D_i$ in Γ where $1 \leq i \leq n$, we have $\sigma(C_i) \sqsubseteq_{\mathcal{T}} \sigma(D_i)$. From Lemma 4.2.1, we know that Γ has an $\mathcal{EL}$ matcher w.r.t. $\mathcal{T}$ iff $\sigma_{\perp(\Gamma,\mathcal{T})}$ is an $\mathcal{EL}$ matcher of Γ w.r.t. $\mathcal{T}$. By construction, $\sigma_{\perp(\Gamma,\mathcal{T})}$ must also satisfy $\sigma_{\perp(\Gamma,\mathcal{T})}(C) \sqsubseteq_{\mathcal{T}} \sigma_{\perp(\Gamma,\mathcal{T})}(D)$. Thus, if σ is an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$, then by construction, $\sigma_{\perp(\Gamma,\mathcal{T})}$ must also an $\mathcal{EL}^{-\top}$-matcher of Γ w.r.t. $\mathcal{T}$. □

The size of $\sigma_{\perp(\Gamma,\mathcal{T})}$ is polynomial in the size of Γ and $\mathcal{T}$, as it only includes symbols existing in Γ and in $\mathcal{T}$. Deciding if a right-ground matching problem in $\mathcal{EL}^{-\top}$ w.r.t. an empty $\mathcal{T}$ has matcher also takes polynomial time, because the construction of $\sigma_{\perp(\Gamma,\mathcal{T})}$ will consider only atoms from $D_1, \ldots, D_n$ as the TBox is empty. Therefore, the complexity of deciding whether a right-ground matching problem has an $\mathcal{EL}^{-\top}$-matcher or not w.r.t. a general $\mathcal{EL}^{-\top}$-TBox remains the same, i.e., in PTime.

Theorem 4.2.4. *Deciding whether a right-ground $\mathcal{EL}^{-\top}$-matching problem modulo subsumption has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. a general $\mathcal{EL}^{-\top}$-TBox takes polynomial time.*

4.2.2 Matching modulo equivalence in $\mathcal{EL}^{-\top}$

In [BK99c; BM14a], matching modulo equivalence for $\mathcal{EL}$ was shown to be NP-complete even in the presence of general $\mathcal{EL}$-TBoxes. NP-hardness of matching modulo equivalence in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox follows from NP-hardness of left-ground matching in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox from theorem 4.1.4. To show that matching modulo equivalence in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox is in NP, we reuse the result from [BK00], where it was shown that matching modulo equivalence for $\mathcal{EL}$ is in NP.

To show the in NP result, it is sufficient to show that every solvable $\mathcal{EL}^{-\top}$ matching problem Γ has an $\mathcal{EL}^{-\top}$ matcher such that,

1. its size is polynomially bounded in the size of Γ, and
2. it uses only concept names and role names already contained in Γ.

To prove the above statement, let us consider n to be the total number

of symbols occurring in Γ. The structure of the concepts in Γ provide a structure to any potential matcher. Therefore, the size of any matcher of Γ is at most n. Any $\mathcal{EL}^{-\top}$ matcher of Γ cannot introduce new concept names or role names that do not already exist in the original matching problem. These conditions can be summarised as the following theorem.

Lemma 4.2.5. *If an $\mathcal{EL}^{-\top}$ matching problem $C \equiv^? D \in \Gamma$ is solvable, then there exists an $\mathcal{EL}^{-\top}$ matcher such that, it is polynomially bound to the size of the matching problem and only uses concept names and role names already contained in Γ.*

We can adapt the proofs from [BK99c] to prove Lemma 4.2.5 above.

As we already know from Theorem 4.1.4, that deciding solvability of a left-ground $\mathcal{EL}^{-\top}$-matching problem modulo subsumption w.r.t. an empty TBox is NP-hard. We also know that left-ground matching is a particular case of matching modulo equivalence. Therefore, Deciding solvability of matching modulo equivalence in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox is in NP. Finally,

Theorem 4.2.6. *Deciding whether a matching problem modulo equivalence in $\mathcal{EL}^{-\top}$ has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. an empty TBox is NP-complete.*

This also leads to the following result.

Theorem 4.2.7. *Deciding whether a left-ground $\mathcal{EL}^{-\top}$-matching problem modulo subsumption has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. an empty TBox is NP-complete.*

Chapter 5

General matching in $\mathcal{EL}^{-\top}$

In [BM14a], deciding whether an $\mathcal{EL}$ matching problem has a matcher w.r.t. to a general $\mathcal{EL}$-TBox was shown to be NP-complete. To show the in NP result, a goal-oriented matching algorithm that uses non-deterministic rules was presented. However, general matching in $\mathcal{EL}^{-\top}$ becomes more difficult as, PSpace-hardness w.r.t. an empty TBox was shown in Chapter 4 [Theorem 4.1.9, 4.1.10]. In this chapter, we will reuse the ideas from the goal oriented matching algorithm and extend it to deal with the restriction of not using $\top$, to solve matching in $\mathcal{EL}^{-\top}$. We show that the complexity of general matching in $\mathcal{EL}^{-\top}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox is in ExpTime, which inturn implies that matching in $\mathcal{EL}^{-\top}$ is in ExpTime.

5.1 The $\mathcal{EL}$ Matching Algorithm

The goal oriented matching algorithm introduced in [BM14a], uses non-deterministic rules to transform a given matching problem Γ into solved form Γ_0 using a polynomial number of rule applications. All offending ground subsumptions are removed without changing the solvability status of the problem. In other words, we can assume that for all subsumptions $C \sqsubseteq^? D$ in Γ_0, either C or D is non-ground. If both are ground we can immediately decide the non-solvability of the problem and the following conditions from [BM14b] hold:

(a) If $C \sqsubseteq_{\mathcal{T}} D$, then Γ_0 has a matcher w.r.t. $\mathcal{T}$ iff $\Gamma_0 \setminus \{C \sqsubseteq^? D\}$ has a matcher w.r.t. $\mathcal{T}$.

1. $Dec(C \sqsubseteq^? D) := \{C \sqsubseteq^? D\}$, if C is a variable.
2. If $D_1, \ldots, D_n$ are atoms, then $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \cdots \sqcap D_n))$ fails if there is an $i \in \{1, \ldots, n\}$ such that both sides of $C' \sqsubseteq^? D_i$ are ground and $C' \not\sqsubseteq_{\mathcal{T}} D_i$. Otherwise, $Dec(\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \cdots \sqcap D_n)) := \{C' \sqsubseteq^? D_i \mid 1 \leq i \leq n$ and C' or D_i is non-ground$\}$.
3. If $C = \exists r.C'$ and $D = \exists s.D'$ for roles $s \neq r$, then $Dec(C \sqsubseteq^? D)$ fails.
4. If $C = A$ is a concept name and $D = \exists r.D'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
5. If $D = A$ is a concept name and $C = \exists r.C'$ an existential restriction, then $Dec(C \sqsubseteq^? D)$ fails.
6. If both C and D are ground and $C \not\sqsubseteq_{\mathcal{T}} D$ then $Dec(C \sqsubseteq^? D)$ fails, and otherwise returns $\emptyset$.

Figure 5.1: The function $Dec(\ldots)$ from [BM14a]

(b) If $C \not\sqsubseteq_{\mathcal{T}} D$, then Γ_0 does not have a matcher w.r.t. $\mathcal{T}$.

This also holds for $\mathcal{EL}^{-\top}$.

The $\mathcal{EL}$ algorithm takes as input a normalized matching problem Γ_0, as shown in 3.3.1. It non-deterministically applies rules to all unsolved subsumptions $C \sqsubseteq^? D$ of Γ_0. An unsolved subsumption is the one that has not been treated or handled by the algorithm during its run. New subsumptions that are not already present in Γ_0 get added, if the application of any rule *succeeds*. Every newly added subsumption is marked as unsolved. Every subsumption that has a variable on either left-hand side or the right-hand side is solved by the application of eager rules as shown in Figure 5.2. Eager rules take precedence over the non-deterministic rules shown in Figure 5.3. Rules are applied exhaustively until all subsumptions are marked as *solved* or a rule application has failed.

The definition of the non-deterministic rules in the algorithm applies the function $Dec(\ldots)$ shown in Figure 5.1, to subsumptions of the form $C \sqsubseteq^? D$, where C and D are atoms and D is not a variable. A call to $Dec(C \sqsubseteq^? D)$ returns a (possibly empty) set of subsumptions or fails. When no more rules are applicable and the algorithm has not returned failure, then it returns success. The algorithm terminates after polynomial number of steps as there are only polynomially many subsumptions that are present in the matching problem Γ. The $\mathcal{EL}$

Eager Solving – variable on the right:

Condition: An unsolved subsumption $C \sqsubseteq^? X \in \Gamma$ where $X \in N_V$.
Action:

- If there is some subsumption of the form $X \sqsubseteq^? D \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails.
- Otherwise, mark $C \sqsubseteq^? X$ as "solved."

Eager Solving – variable on the left:

Condition: An unsolved subsumption $X \sqsubseteq^? D \in \Gamma$ where $X \in N_V$.
Action:

- If there is some subsumption of the form $C \sqsubseteq^? X \in \Gamma$ such that $C \not\sqsubseteq_{\mathcal{T}} D$, then the rule application fails.
- Otherwise, mark $X \sqsubseteq^? D$ as "solved."

Figure 5.2: Eager Rules for $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ from [BM14a]

matching algorithm along with its steps is presented in the Figure 5.4.

Lemma 5.1.1. *If the $\mathcal{EL}$ matching algorithm (in Figure 5.4) has a successful run on an input matching problem Γ, then the induced substitution σ_Γ is an $\mathcal{EL}$ matcher of Γ w.r.t. $\mathcal{T}$.*

To show *soundness* as stated in the lemma above, we show that if the $\mathcal{EL}$ algorithm has a successful run on an input matching problem Γ, then Γ has an $\mathcal{EL}$ matcher. The subsumptions of the form $X \sqsubseteq^? C \in \Gamma$ are used to construct a substitution σ_Γ, such that the conjunction of all the ground concepts C is matcher of Γ. We denote it as $\sqcap S_X^\Gamma$ (as Γ is normalized matching problem implying that C is ground). For each variable $X \in N_V$, we define the set,

$$S_X^\Gamma := \{C \mid X \sqsubseteq^? C \in \Gamma\}.$$

The conjunction of all the elements of S_X^Γ is denoted as $\sqcap S_X^\Gamma$, where the empty conjunction is $\top$. The substitution σ_Γ is defined as,

$$\sigma_\Gamma(X) := \sqcap S_X^\Gamma \text{ for all } X \in N_V.$$

If no subsumptions are present then the empty conjunction $\top$ is considered as the matcher.

To show *completeness*, a matcher σ of Γ_0 w.r.t. $\mathcal{T}$ is used to guide the application of non-deterministic rules towards a non-failing run of the algorithm.

Decomposition:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$

Action: Its application chooses an index $i \in \{1, \ldots, n\}$ and calls $Dec(C_i \sqsubseteq^? D)$. If this call does not fail, then it adds the returned subsumptions to Γ, and marks $\mathfrak{s}$ as solved. If $Dec(C_i \sqsubseteq^? D)$ fails, it returns "failure."

Mutation:

Condition: This rule applies to $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$

Action: Its application tries to choose atoms $A_1, \ldots, A_k, B$ of $\mathcal{T}$ such that $A_1, \sqcap, \cdots, A_k \sqsubseteq_{\mathcal{T}} B$ holds. If this is not possible, then it returns "failure." Otherwise, it performs the following two steps:

- Choose for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ and call $Dec(C_i \sqsubseteq^? A_\eta)$. If this call does not fail, it adds the returned subsumptions to Γ. Otherwise, $Dec(C_i \sqsubseteq^? A_\eta)$ fails, the rule returns "failure."
- If it has not failed before and $Dec(B \sqsubseteq^? D)$ does not fail, it adds the returned subsumptions to Γ. Otherwise, if $Dec(B \sqsubseteq^? D)$ fails, it returns "failure."

If these steps did not fail, then the rule marks $\mathfrak{s}$ as solved.

Figure 5.3: Non-deterministic Rules for $\mathcal{EL}$ and $\mathcal{EL}^{-\top}$ from [BM14a]

Lemma 5.1.2. *[BM14b] Let σ be a matcher of Γ_0 w.r.t. $\mathcal{T}$. Then there is a non-failing and terminating run of Algorithm 5.4 producing a matching problem Γ such that σ is a matcher of Γ w.r.t. $\mathcal{T}$.*

A useful property of the $\mathcal{EL}$ algorithm is mentioned below,

Lemma 5.1.3. *[BM14a] If Γ is a matching problem generated during a non-failing run of the algorithm, and both $C \sqsubseteq^? X \in \Gamma$ and $X \sqsubseteq^? D \in \Gamma$ are solved, then $C \sqsubseteq_{\mathcal{T}} D$.*

5.2 Why does the $\mathcal{EL}$ algorithm not work for $\mathcal{EL}^{-\top}$

In this section, we will first see an example why the $\mathcal{EL}$ algorithm does not always work for $\mathcal{EL}^{-\top}$.

Example 5.2.1. Consider the matching problem Γ,

Input: A normalized $\mathcal{EL}$ matching problem Γ_0 and a general $\mathcal{EL}$-TBox.
Output: *success*, if an $\mathcal{EL}$ matcher exists or *failure*, otherwise.

1. Apply the eager rules and non-deterministic rules exhaustively in the following order,
   - Apply eager rules (if applicable). If an application of an eager rule fails, then *stop* and return *failure*.
   - If no eager rule can be applied, select an unsolved subsumption $\mathfrak{s}$ in Γ. Choose one of the two non-deterministic rules and apply it to $\mathfrak{s}$. If this rule application fails, then *stop* and return *failure*.
2. If no more rules apply and the algorithm has not stopped returning *failure*, return *success*.

Figure 5.4: The $\mathcal{EL}$ matching algorithm
from [BM14a]

$$\Gamma := \{X \sqsubseteq^? \exists r.\top,\ \exists r.A \sqsubseteq^? X\} \text{ and } \mathcal{T} := \{\exists r.A \sqsubseteq \exists r.\top\}.$$

Starting with the first subsumption in Γ, i.e., $X \sqsubseteq^? \exists r.\top$, the eager rule: variable on the left is applied which marks the subsumption as *solved* as there is a subsumption of the form $C \sqsubseteq^? X \in \Gamma$, i.e., $\exists r.A \sqsubseteq^? X$ with $\exists r.A \sqsubseteq_{\mathcal{T}} \exists r.\top$. For the second subsumption, $\exists r.A \sqsubseteq^? X$, the eager rule: variable on the right is applied which is also marked as *solved* as there is a subsumption of the form, $X \sqsubseteq^? D \in \Gamma$, i.e., $X \sqsubseteq^? \exists r.\top$ with $\exists r.A \sqsubseteq_{\mathcal{T}} \exists r.\top$. Since, all subsumptions in Γ are marked as solved, the assignment induced by the successful run of the algorithm using the subsumption of the form $X \sqsubseteq^? D \in \Gamma$, i.e., $X \sqsubseteq^? \exists r.\top$ is $\sigma(X) := \exists r.\top$. Hence, the $\mathcal{EL}$ matcher of Γ is $\exists r.\top$. This example, shows that it is obviously not sound for $\mathcal{EL}^{-\top}$, because the $\mathcal{EL}$ matcher comes from the symbols in the $\mathcal{EL}$ matching problem Γ.

A natural question arises, whether it is possible to slightly modify the algorithm in a way such that the newly obtained algorithm is sound and complete for matching in $\mathcal{EL}^{-\top}$. Is it possible for the algorithm to answer *no* when such an assignment implies the presence of $\top$ in the induced substitution. We will see how this destroys the completeness of the algorithm and show that such a simple modification is not enough to modify the existing $\mathcal{EL}$ matching algorithm into an algorithm that works for $\mathcal{EL}^{-\top}$.

Example 5.2.2. Consider the following matching problem introduced in [Baa+11a] that has both, an $\mathcal{EL}$ matcher and an $\mathcal{EL}^{-\top}$ matcher. We will see how the slightly modified $\mathcal{EL}$ algorithm answers *no* even if an $\mathcal{EL}^{-\top}$ matcher of Γ exists. Let,

$$\Gamma := \{A \sqsubseteq^? X,\ B \sqsubseteq^? X\} \text{ and } \mathcal{T} := \{B \sqsubseteq A\}.$$

Again, starting with the first subsumption in Γ, $A \sqsubseteq^? X$, the eager rule: variable on the right is applied which marks the subsumption as *solved* as there is no subsumption of the form $X \sqsubseteq^? D \in \Gamma$, such that $A \sqsubseteq_{\mathcal{T}} D$. The same applies to the second subsumption, $B \sqsubseteq^? X$ and it is marked as *solved*. To define the substitution σ_Γ, there are no subsumptions of the form $X \sqsubseteq^? D \in \Gamma$. Hence, the empty conjunction or $\top$ is defined as the $\mathcal{EL}$ matcher of Γ. However, Γ has an $\mathcal{EL}^{-\top}$ matcher which is A as $B \sqsubseteq_{\mathcal{T}} A$ and we know $A \sqsubseteq A$. The algorithm answers *no*, as it fails to find the $\mathcal{EL}^{-\top}$ matcher that Γ actually has. Therefore, this modification is not enough to make the $\mathcal{EL}$ algorithm work for an $\mathcal{EL}^{-\top}$ matching problem.

Note that, the substitution $\sigma_\Gamma(X) := A$ can be thought of as a *common subsumer* without $\top$ of A and B as it must satisfy both the subsumption constraints in Γ. The limitation of the $\mathcal{EL}$ matching algorithm to $\mathcal{EL}^{-\top}$ is that it does not always find the $\mathcal{EL}^{-\top}$ matcher that the problem has.

5.3 The $\mathcal{EL}^{-\top}$ Matching Algorithm

In this section, we will show how we can adapt the algorithm to make it work for $\mathcal{EL}^{-\top}$, such that it always answers yes if an $\mathcal{EL}^{-\top}$ matcher exists or returns *failure* otherwise. The $\mathcal{EL}^{-\top}$ matching algorithm as described in figure 5.6 uses the same idea of the $\mathcal{EL}$ algorithm with some additions. From the previous example, given the successful run of the $\mathcal{EL}$ algorithm such that, $X \sqsubseteq^? D \in \Gamma$ is empty

For an $\mathcal{EL}^{-\top}$ matching problem Γ, if the algorithm answers *yes* implying that Γ has an $\mathcal{EL}$ matcher, the subsumptions of the form $X \sqsubseteq^? D \in \Gamma$ are used. For every variable $X \in N_V$, the substitution $\sigma_\Gamma(X) := \sqcap S_X^\Gamma$ is defined where, $S_X^\Gamma := \{D \mid X \sqsubseteq^? D \in \Gamma\}$. For the case where an empty conjunction or $\top$ gets assigned to a variable as there are no subsumptions of the form $\{X \sqsubseteq^? D \in \Gamma\}$ and the set $S_X^\Gamma = \emptyset$, a common

subsumer not containing $\top$ needs to be assigned to X as the $\mathcal{EL}^{-\top}$ matcher of Γ. This is how we extend the $\mathcal{EL}$ algorithm into a new algorithm for $\mathcal{EL}^{-\top}$. More precisely, we introduce a new rule called the *common subsumer rule* that decides the existence of common subsumers without $\top$. The *common subsumer rule* extends the $\mathcal{EL}$ matching algorithm to decide general matching in $\mathcal{EL}^{-\top}$, as shown in Figure 5.5. We can assume the concepts assigned to the variables $X \in N_V$ to be particles as we will show in the next section.

Common subsumer rule:

Condition: For any variable $X \in N_V$ such that, there exists at least one subsumption of the form $C \sqsubseteq^? X \in \Gamma$ and no subsumptions of the form $X \sqsubseteq^? D \in \Gamma$.

Action: Let $\{C_1, C_2, \ldots, C_n\}$ be the set of concepts such that $C_i \sqsubseteq^? X \in \Gamma$ with $1 \leq i \leq n$.

- If a common subsumer without $\top$ of $\{C_1, C_2, \ldots, C_n\}$ w.r.t. $\mathcal{T}$ exists, then return "success".
- Otherwise, return "failure".

Figure 5.5: Common subsumer rule for $\mathcal{EL}^{-\top}$

Let us see if the new algorithm works for the Example 5.2.2. When $\Gamma := \{A \sqsubseteq^? X,\ B \sqsubseteq^? X\}$ and $\mathcal{T} := \{B \sqsubseteq A\}$, the $\mathcal{EL}$ algorithm was successful in applying all the rules therefore marking all subsumption constraints as *solved*. There are no subsumptions of the form $X \sqsubseteq^? C \in \Gamma$. But, in the $\mathcal{EL}^{-\top}$ matching algorithm, we can apply the common subsumer rule since the set $S_X^\Gamma = \emptyset$ and we have the subsumptions of the form $C_1 \sqsubseteq^? X$ and $C_2 \sqsubseteq^? X$ in Γ which are $A \sqsubseteq^? X$ and $B \sqsubseteq^? X$ and we know that $B \sqsubseteq_\mathcal{T} A$. The common subsumer not containing $\top$ is A. Therefore, the $\mathcal{EL}^{-\top}$ matcher of the problem is the concept name A as $A \sqsubseteq A$ and $B \sqsubseteq_\mathcal{T} A$. If a common subsumer without $\top$ does not exist then the algorithm returns *failure* indicating that an $\mathcal{EL}^{-\top}$ matcher does not exist, which also implies that there is no $\mathcal{EL}$ matcher of Γ because an $\mathcal{EL}^{-\top}$ matcher is also an $\mathcal{EL}$ matcher.

The new $\mathcal{EL}^{-\top}$ algorithm consists of an additional rule to decide the existence of a common subsumer without $\top$. In the next section, we will describe methods on how such a common subsumer without $\top$ can be computed with the help of notions introduced in the Section 2.3. Having introduced the common subsumer rule in figure 5.5, we will introduce a

Input: A normalized $\mathcal{EL}^{-\top}$ matching problem Γ_0 and a general $\mathcal{EL}^{-\top}$ – $TBox$.
Output: *success*, if an $\mathcal{EL}^{-\top}$ exists or *failure*, otherwise.

1. Apply the eager rules and non-deterministic rules exhaustively in the following order,
   - Apply eager rules (if applicable). If an application of an eager rule fails, then *stop* and return *failure*.
   - If no eager rule can be applied, select an unsolved subsumption $\mathfrak{s}$ in Γ. Choose one of the two non-deterministic rules and apply it to $\mathfrak{s}$. If this rule application fails, then *stop* and return *failure*.
2. If no more rules apply and the algorithm has not stopped returning *failure*, proceed to step 3.
3. Apply the common subsumer rule for all $X \in N_V$ in Γ_0.
4. If no rule application fails, return *success*.

Figure 5.6: The $\mathcal{EL}^{-\top}$ matching algorithm

decision procedure for the existence of common subsumer(s) without $\top$ in the next section. The proof for the correctness of the $\mathcal{EL}^{-\top}$ matching algorithm is deferred to section 5.5.

5.4 Decision procedure for the existence of common subsumers without $\top$

For a set of concept descriptions, a *common subsumer (cs)* is a concept description that subsumes all elements from the set. In this chapter, we present an algorithm that decides whether two concepts C and D have a common subsumer without $\top$ w.r.t. a TBox $\mathcal{T}$. The algorithm can also be extended to a sequence of concepts $C_1, \ldots, C_m$. We will show how this can be achieved towards the end of this section.

Definition 5.4.1. Let C, D be $\mathcal{EL}^{-\top}$ concepts and $\mathcal{T}$ be a *general* $\mathcal{EL}^{-\top}$-TBox. A concept E is a *common subsumer* without $\top$ of C and D w.r.t. $\mathcal{T}$, if both the properties are satisfied:

1. $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$, and
2. E does not contain $\top$.

The set of common subsumers without $\top$ of two concepts is denoted as $cs_{\mathcal{T}}^{-\top}(C, D)$. If there is another concept F that satisfies the properties above, i.e., $C \sqsubseteq_{\mathcal{T}} F$ and $D \sqsubseteq_{\mathcal{T}} F$ and F does not contain $\top$, then it belongs to this set, as common subsumers are not unique unlike *least common subsumers* [ZT13a], because being an *lcs* means it is the most specific concept that is subsumed all other common subsumers and common subsumers without $\top$ need not satisfy this property. We will use a short-hand notation *"common subsumers or (cs)"* to refer to a common subsumer of C and D w.r.t. $\mathcal{T}$ that does not contain $\top$ for the rest of the section.

If C is a common subsumer without $\top$ and has depth at most k, then we say that C belongs to the set of common subsumers of role depth at most k.

To decide the existence of common subsumers without $\top$ w.r.t. general $\mathcal{EL}^{-\top}$-TBoxes, the correctness of the computation algorithm depends on the notion of canonical models and simulation relations introduced in Section 2.3 [ZT13b].

We adapt the ideas from [ZT13a] to decide the existence of a least common subsumer. In order to decide the existence of a common subsumer without $\top$ of two concepts w.r.t. to a TBox, we use the following steps.

- *Identify candidates for common subsumers without* $\top$.
  The set of common subsumers without $\top$ is a possibly infinite set of common subsumers of C and D w.r.t. $\mathcal{T}$.
- *Characterize particles as common subsumers without* $\top$.
  If a common subsumer without $\top$ exists then we prove there exists a particle that is common subsumer without $\top$.

To obtain the candidates for common subsumers without $\top$, we build the product of canonical models $(\mathcal{I}_{C,\mathcal{T}}, d_C)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D)$ and construct the k-characteristic concept of this product model as introduced in Section 2.3.3 and also shown in [ZT13b] [LW10]. The k-characteristic concept captures all commonalities of the two input concepts C and D, hence it is considered as a suitable candidate.

Lemma 5.4.2. *[ZT13b] Let k be a natural number ($k \in \mathbb{N}$).*

1. $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \in cs_{\mathcal{T}}(C, D)$.

2. *Let E be a concept with $rd(E) \leq k$ and $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$. It holds that $X^k(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)) \sqsubseteq_{\mathcal{T}} E$.*

The lemma above states that every common subsumer E, subsumes the k-characteristic concept (where $rd(E) \leq k$). The set the all k-characteristic concepts of the product model belongs to the set of common subsumers of C and D w.r.t. $\mathcal{T}$ and we can find another concept E of role-depth $\leq k$ that subsumes the k-characteristic concept and is also a common subsumer of C and D w.r.t. $\mathcal{T}$. From this result in Lemma 5.4.2, we can say that we have a *common subsumer* without $\top$ if either one of the k-characteristic concepts from the set does not contain $\top$ or there is a concept E that subsumes it and does not contain $\top$.

Lemma 5.4.3. *Let C and D be two $\mathcal{EL}^{-\top}$ concepts and $\mathcal{T}$ an $\mathcal{EL}^{-\top}$-TBox. If C and D have a common subsumer without $\top$ w.r.t. $\mathcal{T}$, then we have a common subsumer E that is a particle.*

Proof. Assume that C and D have a common subsumer E without $\top$ w.r.t. $\mathcal{T}$. Then, E is a particle as it does not contain $\top$. From Lemma 2.2.2 we can say that these particles subsume the concept E. Therefore, to decide the existence of a common subsumer without $\top$, we can restrict our attention to particles, i.e., if a concept that is a common subsumer without $\top$ of C and D exists then we can assume this concept to be a particle (see Definition 2.1.2). □

First, it is sufficient to check whether $\top$ occurs in the k-characteristic concept, because if it does not we can say that it yields a common subsumer without $\top$. But, in the case where $\top$ occurs, we cannot say that a common subsumer without $\top$ does not exist. We need to find another concept E that subsumes the characteristic concept and does not contain $\top$ as shown in lemma 5.4.2.

We can assume such a concept E to be a particle from Lemma 5.4.3. To decide whether a common subsumer without $\top$ exists we can restrict the search to particles that subsume the k-characteristic concept. The next step is to show that if such a particle E exists, then E must be a particle of the characteristic concept of the same role-depth as E, i.e., $Part(X^k(\mathcal{I}, d))$ of role depth k such that $rd(E) \leq k$. This check is more stronger and we can prove that if such a concept (now a particle) exists then it yields a common subsumer without $\top$ of C and D w.r.t. $\mathcal{T}$.

Let $(\mathcal{I}, d)$ be the product graph of canonical models of C and D and $X^k(\mathcal{I}, d)$ the k-characteristic concept of this product model with role depth at most k. We can say the following,

Lemma 5.4.4. *Let C and D be two $\mathcal{EL}^{-\top}$ concepts, E be a particle and $\mathcal{T}$ be a general $\mathcal{EL}^{-\top}$-TBox. Then, we have E is a common subsumer of C and D w.r.t. $\mathcal{T}$ iff $E \in Part(X^k(\mathcal{I}, d))$ with $rd(E) \leq k$.*

Proof. ($\Rightarrow$) For this direction, assume that E is a common subsumer of C and D w.r.t. $\mathcal{T}$. We show that E belongs to the set of particles of the k-characteristic concept with $k = rd(E)$.

From definition 5.4.1 we have $C \sqsubseteq_{\mathcal{T}} E$ and $D \sqsubseteq_{\mathcal{T}} E$ and from Lemma 5.4.2 we know that the k-characteristic concept of C and D w.r.t. $\mathcal{T}$ is subsumed by E. Applying the Lemma 2.2.4 to $C \sqsubseteq_{\mathcal{T}} E$, we can say that the element is the root of the canonical model of C w.r.t. $\mathcal{T}$. This implies it also belongs to the interpretation E in $\mathcal{I}_{C,\mathcal{T}}$, i.e., $d_C \in E^{\mathcal{I}_{C,\mathcal{T}}}$. The same applies to the concept D, i.e., $d_D \in E^{\mathcal{I}_{C,\mathcal{T}}}$.

The canonical models of C and D must have a path each from the root d_C and d_D respectively such that the root is an instance of the particle E. From the construction of the product model as shown in 2.3.3, we can say that the product of the canonical models of C and D must have a path from the root (d_C, d_D) such that the root is an instance of the particle E. The construction of the characteristic concept of depth k is based on the unraveling of the product model 2.3.4. If there exists a path of depth k in the product model then this path must also exist in the tree unraveling of this product model. We construct the characteristic concept from the tree unraveling as shown in definition 2.3.5. The existence of such a path in the unraveling implies the existence of the concept E that is particle. Therefore, we can say that the concept $E \in Part(X^k(\mathcal{I}, d))$ with $rd(E) = k$.

($\Leftarrow$) For the right to left direction, assume E is a particle that belongs to the set $Part(X^k(\mathcal{I}, d))$ with $rd(E) = k$. From Lemma 2.2.2, we can say that if $E \in Part(X^k(\mathcal{I}, d))$ then $X^k(\mathcal{I}, d) \sqsubseteq E$. This result only holds w.r.t. to an empty TBox. However, we can say that it also holds w.r.t. a general $\mathcal{EL}^{-\top}$-TBox because every subsumption that is true w.r.t. to an empty TBox is also true w.r.t. a general $\mathcal{EL}^{-\top}$-TBox. From Lemma 5.4.2 we can say that if concept E subsumes the characteristic concept of same depth then it also belongs to the set of common

Input: Let C and D be two $\mathcal{EL}^{-\top}$ concepts, and $\mathcal{T}$ a general $\mathcal{EL}^{-\top}$-TBox.
Output: *yes*, if a common subsumer without $\top$ of C and D w.r.t. $\mathcal{T}$ exists or *no*, otherwise.

1. Build the product graph of canonical models $(\mathcal{I}_{C,\mathcal{T}}, d_C)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D)$, denoted as $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$.
2. Check if there exists a path in the product graph from the root (d_C, d_D) to an element labelled with a concept name.
3. If a path is found in Step 2 then return *yes*, otherwise return *no*.

Figure 5.7: Decision procedure for the existence of *common subsumers* without $\top$

subsumers without $\top$ implying that E which is a particle is also a common subsumer of C and D w.r.t. $\mathcal{T}$. This completes the proof in both directions. □

The previous lemma tells us that we only need to check whether some k-characteristic concept has a particle. What remains to be shown is that if a common subsumer without $\top$ exists then its role-depth is bounded by the size of the product model. In other words, to decide if such a particle exists.

If a particle $\exists w.A$ is a common subsumer then the product model has a path from the root to a node labelled with concept name A. In order to decide whether a particle is a common subsumer without $\top$, it is sufficient to find a path in the product model. However, the product model of two concepts is finite graph containing cycles. It is easy to find such a path in the product graph by visiting every node, if a common subsumer without exists. In case where it does not, we can characterize every node as *"visited"* and answer *no* when every node has been visited in the process of finding such a path to a particle.

Next, we present an algorithm in 5.7 to decide the existence of common subsumers without $\top$ of two $\mathcal{EL}^{-\top}$concepts C and D w.r.t. $\mathcal{T}$. In step 1, building the product model takes polynomial time. In step 2, checking the existence if a path in the product model which is finite can be done in polynomial time. The algorithm terminates when it has found a particle in the set $Part(X^{\ell}(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)))$. Hence, we can say that the algorithm to decide the existence of a common subsumer without $\top$ of two $\mathcal{EL}^{-\top}$concepts C and D takes polynomial time.

To show soundness, we show that the algorithm returns *"yes"* only when a common subsumer of C and D without $\top$ w.r.t. $\mathcal{T}$ indeed exists. We will go through each step of the algorithm to show that it answers yes when a concept E that is a particle exists. The algorithm constructs the ℓ-characteristic concepts $(X^\ell(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)))$ of the product model $(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D))$ using lemma 2.3.5, from $\ell = \{0, \ldots, n\}$. Since the algorithm returned *"yes"*, it finds a particle E in the set $Part(X^\ell(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)))$ for some role depth ℓ where $rd(E) = \ell$. From lemma 5.4.4 we can say that E is a common subsumer of C and D without $\top$ w.r.t. $\mathcal{T}$. Therefore, the algorithm answers *"yes"* which shows that the algorithm is indeed sound.

To show completeness, we show that the algorithm returns *"yes"* whenever a common subsumer without $\top$ exists. We know that if the algorithm answers yes, then there exists a particle or concept E that is a common subsumer of C and D w.r.t. $\mathcal{T}$. From lemma 5.4.4 there exists a k such that this concept is a particle in $\text{Part}(X^\ell(\mathcal{I}_{C,\mathcal{T}} \times \mathcal{I}_{D,\mathcal{T}}, (d_C, d_D)))$. Therefore, the algorithm answers *"yes"* only when it finds a common subsumer of C and D without $\top$ w.r.t. $\mathcal{T}$.

Theorem 5.4.5. *Let C and D be two $\mathcal{EL}^{-\top}$ concepts and $\mathcal{T}$ be a general $\mathcal{EL}^{-\top}$-TBox. Deciding if C and D have a common subsumer without $\top$ w.r.t. $\mathcal{T}$ can be done in polynomial time.*

The results shown above can be easily generalized to finding common subsumers without $\top$ of an arbitrary set of concepts $M = \{C_1, \ldots, C_m\}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox $\mathcal{T}$. In this general case, we have to take the product model

$$(\mathcal{I}_{C_1,\mathcal{T}} \times \cdots \times \mathcal{I}_{C_m,\mathcal{T}}, (d_{C_1}, \cdots, d_{C_m})),$$

whose size is exponential in the number of concepts in M and $\mathcal{T}$. Then the same steps as for the binary version can be applied.

Theorem 5.4.6. *Deciding the existence of a common subsumer without $\top$ of m concepts can be done in exponential time in the size of the number of concepts in M and $\mathcal{T}$.*

We will prove the correctness of the $\mathcal{EL}^{-\top}$ matching algorithm in the next section using the decision procedure for common subsumers without $\top$ in Figure 5.7.

5.5 Correctness of the $\mathcal{EL}^{-\top}$ matching algorithm

5.5.1 Soundness

To prove *soundness*, we show that if the $\mathcal{EL}^{-\top}$ matching algorithm answers *yes*, then the input matching problem Γ has an $\mathcal{EL}^{-\top}$ matcher σ. If the algorithm answers yes, then it has a successful run and every successful run induces a substitution σ_Γ.

Lemma 5.5.1. *Let Γ be an $\mathcal{EL}^{-\top}$ matching problem obtained after a terminating and non-failing of the algorithm 5.3 and $\mathcal{T}$ be a general $\mathcal{EL}^{-\top}$-TBox. Then, we can say that σ_Γ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. We have already seen in the Lemma 5.4.4, that if a common subsumer without $\top$ exists then there exists a particle that is a common subsumer without $\top$. A particle is of the form $\exists w.A$ (see Definition 2.1.2) and it does not contain $\top$. Therefore, we can say that the if a variable $X \in N_V$ in Γ is mapped to a particle that is a common subsumer, then it does not contain $\top$.

Next, we show that the substitution σ_Γ is obtained after the successful and terminating run of the algorithm is indeed an $\mathcal{EL}^{-\top}$ matcher of the original matching problem Γ_0. To prove this we need to show that σ_Γ actually solves all the subsumptions of the problem Γ. We go through each type of subsumption and see how σ_Γ solves them. The proof idea is very similar to the proof described in [BM14b], but now we have two new conditions: (a) $\top$ is not allowed and (b) We have a new rule to find the common subsumer(s) without $\top$ for subsumptions of the form $C_i \sqsubseteq^? X$.

- Subsumptions of the form $X \sqsubseteq^? D$ are solved by σ_Γ since $\sigma_\Gamma(X) \sqsubseteq_\mathcal{T} D$. This is true because D is a conjunct of $\sigma_\Gamma(X)$.
- For subsumptions of the form $C \sqsubseteq^? X$, by definition if $S^\Gamma_X \neq \emptyset$ then $\sigma_\Gamma(X) = D_1 \sqcap \cdots \sqcap D_n$ where for each $i \in \{1, \ldots, n\}$, $X \sqsubseteq^? D_i \in \Gamma$. Given that the algorithm has successfully terminated with the final matching problem Γ, the subsumption $C \sqsubseteq^? X$ as well as all subsumptions $X \sqsubseteq^? D_i$ are marked as solved in Γ. According to Lemma 5.1.3, this implies $C \sqsubseteq_\mathcal{T} D_i$ for all $i \in$

$\{1, \ldots, n\}$, which implies $C \sqsubseteq_{\mathcal{T}} D_1 \sqcap \cdots \sqcap D_n = \sigma_\Gamma(X)$. Now, if $S_X^\Gamma = \emptyset$ then we have $C_i \sqsubseteq^? X \in \Gamma$ where $i \in \{1, \ldots, n\}$. From the common subsumer rule if a common subsumer exists then, $cs(C_i) \sqsubseteq \sigma_\Gamma(X)$. Hence, $\sigma_\Gamma(X)$ solves $C_i \sqsubseteq^? X$.

- To show that σ_Γ solves other subsumptions $\mathfrak{s} = C \sqsubseteq^? D$ in Γ, we use induction over the size of $\mathfrak{s}$. The size of $\mathfrak{s}$ is defined as $|C|$ if C is non-ground and $|D|$ if D is non-ground. The idea is that applying a non-deterministic rule to a subsumption $\mathfrak{s}$ generates new subsumptions where the size of the non-ground side is smaller than that of the non-ground size of $\mathfrak{s}$. Subsumptions with a non-ground side of size 1 which are of the form $C \sqsubseteq^? X$ or $X \sqsubseteq^? D$, have already been addressed above. For larger subsumptions, they are solved by either Mutation or Decomposition. In case of Decomposition, consider a subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$ solved by this rule. For some $i \in \{1, \ldots, n\}$, the call $Dec(C_i \sqsubseteq^? D)$ is computed. Since this call does not fail, the cases 1,2 and 6 from the definition of Dec applies,

  - Case 1: C_i is a variable and $Dec(C_i \sqsubseteq^? D) = \{C_i \sqsubseteq^? D\}$. Thus, $C_i \sqsubseteq^? D \in \Gamma$ and we have already seen that σ_Γ solves such subsumptions, i.e., $\sigma_\Gamma(C_i) \sqsubseteq_{\mathcal{T}} D$. This implies that σ_Γ also solves $C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D$.

  - Case 2: $C_i = \exists r.C'$ and $D = \exists r.(D_1 \sqcap \cdots \sqcap D_m)$. Since Dec did not fail, we have $C_i \sqsubseteq_{\mathcal{T}} D_j$ for all subsumptions $C_i \sqsubseteq^? D_j$, where both sides are ground. Non-ground subsumptions $C_i \sqsubseteq^? D_j$ are added to Γ and are smaller than $\mathfrak{s}$. Thus, σ_Γ solves these subsumptions which implies that it solves $\mathfrak{s}$ as well.

  - Case 6: Both C_i and D are ground, and $C_i \sqsubseteq_{\mathcal{T}} D$. Since, Dec did not fail. This implies that σ_Γ solves $\mathfrak{s}$.

- In case of Mutation, assume the subsumption $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$ is solved by this rule. Since the application of this rule does not fail, there exist atoms $A_1, \ldots, A_k$, B in $\mathcal{T}$ such that $A_1, \sqcap \cdots \sqcap A_k \sqsubseteq_{\mathcal{T}} B$. The rule chooses for each $\eta \in \{1, \ldots, k\}$ an $i \in \{1, \ldots, n\}$ such that none of the calls $Dec(C_i \sqsubseteq^? A_\eta)$ fails. Additionally, it calls $Dec(B \sqsubseteq^? D)$, and this call does not fail either. Similar to the Decomposition rule, we can show that

all subsumptions $C_i \sqsubseteq^? A_\eta$ and $B \sqsubseteq^? D$ are solved by σ_Γ. This implies that σ_Γ also solves $\mathfrak{s}$.

The input matching problem Γ_0 is contained in Γ and the proof above shows that σ_Γ is a matcher of Γ which implies that it also a matcher of Γ_0 w.r.t. $\mathcal{T}$. This completes the proof of soundness. □

5.5.2 Completeness

To prove the completeness of the algorithm we consider σ as the $\mathcal{EL}^{-\top}$ matcher of Γ_0 w.r.t. a TBox $\mathcal{T}$ and guide the algorithm towards a successful and terminating run such that σ is also a matcher of Γ w.r.t. $\mathcal{T}$. The following lemma states the claim.

Lemma 5.5.2. *Let σ be a an $\mathcal{EL}^{-\top}$ matcher of Γ_0 w.r.t. a TBox $\mathcal{T}$. Then there is a non-failing and terminating run of algorithm 5.6 producing a matching problem Γ such that σ is a matcher of Γ w.r.t. $\mathcal{T}$.*

Proof. We show that the rule applications are performed on Γ without failure while maintaining the following invariant,

(**Inv:**) *All subsumptions in the matching problem Γ generated during the execution are solved by σ.* [BM14b]

Initially, for the matching problem $\Gamma = \Gamma_0$ this invariant is satisfied since σ was assumed to be a matcher of Γ_0 with respect to $\mathcal{T}$. The application of an eager rule cannot fail since σ solves the subsumptions involved. If Γ contains the subsumption $C \sqsubseteq^? X$ and $X \sqsubseteq^? D$, then by the invariant these subsumptions are solved by σ and thus $C \sqsubseteq_\mathcal{T} \sigma(X) \sqsubseteq_\mathcal{T} D$, which by the transitivity of subsumption yields $C \sqsubseteq_\mathcal{T} D$. The non-deterministic rules use the function *Dec* to solve the subsumptions and we rely on the correctness of the following claim:

Claim: *If σ structurally solves the subsumption $C \sqsubseteq^? D$ (i.e., $\sigma(C) \sqsubseteq^s_\mathcal{T} \sigma(D)$), then the call $Dec(C \sqsubseteq^? D)$ does not fail and σ solves all the subsumptions returned by this call.* [BM14b]

To prove this claim, we consider each case of the *Dec* function as described in figure 5.1.

- Case 1 is trivial as it never fails and return the input subsumption.

- In case 2, σ structurally solves $\exists r.C' \sqsubseteq^? \exists r.(D_1 \sqcap \cdots \sqcap D_n)$ implies that $\sigma(C') \sqsubseteq_\mathcal{T} \sigma(D_i)$ for all $i \in \{1, \ldots, n\}$. This implies $C' \sqsubseteq_\mathcal{T} D_i$ for indices i where both C' and D_i are ground, ensuring the call does not fail. Additionally, σ also solves the returned subsumptions $C' \sqsubseteq^? D_i$ where one side is non-ground.
- Cases 3 and 4 do not apply because σ cannot structurally solve the respective subsumption.
- In case 6, if $C \sqsubseteq_\mathcal{T} D$, then σ must solve the ground subsumption $C \sqsubseteq^? D$. This proves the claim.

To handle unsolved subsumptions $\mathfrak{s} = C_1 \sqcap \cdots \sqcap C_n \sqsubseteq^? D \in \Gamma$ where no eager rule can be applied and neither D nor C_1 if n = 1 is a variable, and we know $\sigma(C_1) \sqcap \cdots \sqcap \sigma(C_n) \sqsubseteq_\mathcal{T} \sigma(D)$ holds, there are three possibilities that may justify the subsumption hierarchy.

- if $n > 1$ and there is an index $i \in \{1, \ldots, n\}$ such that C_i is a variable and $\sigma(C_i) \sqsubseteq_\mathcal{T} \sigma(D)$. The algorithm can apply Decomposition to $\mathfrak{s}$ and call $Dec(C_i \sqsubseteq^? D)$. Since C_i is a variable, this call does not fail and returns $(C_i \sqsubseteq^? D)$, which is added to Γ preserving the invariant.
- If there is an index $i \in \{1, \ldots, n\}$ such that C_i is not a variable and $\sigma(C_i) \sqsubseteq^s_\mathcal{T} \sigma(D)$. Here, we apply Decomposition, selecting an index i, and call $Dec(C_i \sqsubseteq^? D)$. Since σ structurally solves $(C_i \sqsubseteq^? D)$ the call does not fail and returns subsumptions solved by σ.
- If there exist atoms $A_1, \ldots, A_k$, B of $\mathcal{T}$ such that $A_1 \sqcap \cdots \sqcap A_k \sqsubseteq_\mathcal{T} B$. For each $\eta \in \{1, \ldots, k\}$ there is an $i \in \{1, \ldots, n\}$ such that $\sigma(C_i) \sqsubseteq^s_\mathcal{T} A_\eta$ and $B \sqsubseteq^s_\mathcal{T} \sigma(D)$. In this case, apply Mutation, selecting these atoms $A_1, \ldots, A_k$, B. Since, σ structurally solves $C_i \sqsubseteq^? A_\eta$ and $B \sqsubseteq^? D$, the calls to Dec in the rule does not fail and produce only subsumptions solved by σ. Thus, mutation rule preserves the invariant.

If Γ contains subsumptions of the form $C_i \sqsubseteq^? X$ for all $i \in \{1, \ldots, n\}$, the application of the common subsumer rule does not fail as $cs(\sigma(C_i)) \sqsubseteq^? X \sqsubseteq_\mathcal{T} D$, thereby preserving the invariant.

To sum up, the application of eager rules does not fail and maintains the invariant. If no eager rule is applicable and an unsolved subsumption

remains then a non-deterministic rule can be applied that solves the subsumption ensuring no failure and preservation of the invariant. The application of the common subsumer rule then checks for the existence of a common subsumer without $\top$ to σ. Consequently, the algorithm has a successful run where σ is a matcher w.r.t. $\mathcal{T}$ for the final matching problem Γ generated during this run. This finishes the proof of *completeness*. □

5.5.3 Termination and Complexity

Any rule application, as illustrated in Figure: 5.6, either fails while attempting to solve an unsolved subsumption (causing the algorithm to terminate immediately) or successfully solves an unsolved subsumption. Hence, there are only polynomial number of rule applications during a run. It is also easy to see that the eager rule and non-deterministic rule application can be realised in polynomial time, with only a polynomial number of possible non-deterministic choices. However, deciding the existence a common subsumer without $\top$ of n concepts can take exponential time.

As we already know that the complexity of the $\mathcal{EL}$ matching algorithm is in NP and the $\mathcal{EL}^{-\top}$ matching algorithm uses the same idea with an additional component to decide the existence of common subsumers for $n \in \mathbb{N}$ concepts.[1]. Since the $\mathcal{EL}^{-\top}$ matching algorithm 5.3 may contain $n \in \mathbb{N}$, subsumptions $C_1 \sqsubseteq^? X, \ldots, C_n \sqsubseteq^? X$, the computational complexity now depends on the complexity of deciding the existence of common subsumers for $n \in \mathbb{N}$ concepts which takes exponential time. We can conclude that our new algorithm to decide matching in $\mathcal{EL}^{-\top}$ runs in exponential time. This implies that general matching in $\mathcal{EL}^{-\top}$ can be decided in exponential time.

Theorem 5.5.3. *Deciding whether a general $\mathcal{EL}^{-\top}$ matching problem has an $\mathcal{EL}^{-\top}$ matcher or not w.r.t. a general $\mathcal{EL}^{-\top}$-TBox takes exponential time.*

[1]Note that the Algorithm 5.7 for common subsumers without $\top$ only considers two input concepts. It can be adapted to n concepts as demonstrated at the end of Section 5.4.1

Chapter 6

Conclusions and Future work

In this thesis, we have investigated the computational complexity of matching in $\mathcal{EL}^{-\top}$, both in the case of an empty TBox and when a general $\mathcal{EL}^{-\top}$-TBox is present. We considered four variants of the matching problem for $\mathcal{EL}^{-\top}$ introduced in Section 3.2 and obtained the following complexity results.

- PTime complexity of right-ground matching modulo subsumption w.r.t. arbitrary TBoxes in Section 4.2.1.
- Deciding whether a left-ground matching problem modulo subsumption in $\mathcal{EL}^{-\top}$ has a matcher (w.r.t. an empty TBox) was shown to be NP-complete, as discussed in Section 4.1, while the presence of a general $\mathcal{EL}^{-\top}$-TBox increases the complexity to PSpace-hard, as shown in Theorem 4.1.9.
- Matching modulo equivalence in $\mathcal{EL}^{-\top}$ w.r.t. an empty TBox was also shown to be NP-complete. However, in the presence of a general $\mathcal{EL}^{-\top}$-TBox it becomes PSpace-hard, as demonstrated in Section 4.2.2.
- Lastly, we showed that general matching in $\mathcal{EL}^{-\top}$ w.r.t. a general $\mathcal{EL}^{-\top}$-TBox is decidable, with complexity between PSpace-hard and in ExpTime. The new algorithm presented to decide matching in $\mathcal{EL}^{-\top}$ is based on extending the $\mathcal{EL}^{-\top}$ matching algorithm from [BM14a] with the algorithm to decide the existence of common subsumers without $\top$ as shown in Section 5.4.1.

The complexity results of matching in $\mathcal{EL}^{-\top}$ significantly differ from the

	$\mathcal{EL}$	
	$\mathcal{T} = \emptyset$	$\mathcal{T} \neq \emptyset$
Left-ground matching modulo subsumption	P	P
Right-ground matching modulo subsumption	P	P
Matching modulo equivalence	NP-complete	NP-complete
General matching	NP-complete	NP-complete

Table 6.1: Complexity results for $\mathcal{EL}$ from [BK00; BM14a]

	$\mathcal{EL}^{-\top}$	
	$\mathcal{T} = \emptyset$	$\mathcal{T} \neq \emptyset$
Left-ground matching modulo subsumption	NP-complete	PSpace-hard, in ExpTime
Right-ground matching modulo subsumption	P	P
Matching modulo equivalence	NP-complete	PSpace-hard, in ExpTime
General matching	NP-hard, in ExpTime	PSpace-hard, in ExpTime

Table 6.2: Complexity results for $\mathcal{EL}^{-\top}$

ones obtained for $\mathcal{EL}$. Therefore, when $\top$ is not allowed in a description logic under consideration, matching becomes more challenging. The results are summarized in Tables 6.1 and 6.2.

As future work, demonstrating that general matching in $\mathcal{EL}^{-\top}$ is in PSpace would yield a matching upper bound. This will close the gap in the complexity of the three variants of matching, i.e., left-ground matching, matching modulo equivalence and general matching w.r.t. arbitrary TBoxes. Hence, it would allow us to conclude PSpace-completeness for all these variants. For the open gap for general matching w.r.t. an empty TBox, whose computational complexity currently lies between NP-hard and in ExpTime, it would be interesting to see if an NP algorithm could be used to solve this problem resulting in NP-completeness for this case. Additionally, exploring practical applications of the new $\mathcal{EL}^{-\top}$ matching algorithm would would be an interesting and valuable direction.

Bibliography

[Koz77] Dexter Kozen. "Lower bounds for natural proof systems". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE. 1977, pp. 254–266.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979.

[BM96] Alexander Borgida and Deborah L McGuinness. "Asking Queries about Frames." In: *KR* 96 (1996), pp. 340–349.

[BK99a] F. Baader and R. Küsters. *Matching Concept Descriptions with Existential Restrictions Revisited*. LTCS-Report LTCS-99-13. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html. Germany: LuFG Theoretical Computer Science, RWTH Aachen, 1999. DOI: https://doi.org/10.25368/2022.98.

[BK99b] F. Baader and R. Küsters. *Matching in Description Logics with Existential Restrictions*. LTCS-Report LTCS-99-07. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html. Germany: LuFg Theoretical Computer Science, RWTH Aachen, 1999. DOI: https://doi.org/10.25368/2022.93.

[BK99c] F. Baader and R. Küsters. "Matching in Description Logics with Existential Restrictions". In: *Proceedings of the International Workshop on Description Logics 1999 (DL'99)*. Ed. by P. Lambrix et al. CEUR-WS 22. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-22/
. Sweden: Linköping University, 1999.

[BKM99] F. Baader, R. Küsters, and R. Molitor. "Computing Least Common Subsumers in Description Logics with Existential Restrictions". In: *Proceedings of the 16th International*

Joint Conference on Artificial Intelligence (IJCAI'99). Ed. by T. Dean. Morgan Kaufmann, 1999, pp. 96–101.

[Baa+99] F. Baader et al. "Matching in Description Logics". In: *Journal of Logic and Computation* 9.3 (1999), pp. 411–447.

[BK00] F. Baader and R. Küsters. "Matching in Description Logics with Existential Restrictions". In: *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR2000)*. Ed. by A.G. Cohn, F. Giunchiglia, and B. Selman. San Francisco, CA: Morgan Kaufmann Publishers, 2000, pp. 261–272.

[BN01] F. Baader and P. Narendran. "Unification of Concepts Terms in Description Logics". In: *J. Symbolic Computation* 31.3 (2001), pp. 277–305.

[Baa02] F. Baader. *Terminological Cycles in a Description Logic with Existential Restrictions*. LTCS-Report LTCS-02-02. See http://lat.inf.tu-dresden.de/research/reports.html. Germany: Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, 2002. DOI: https://doi.org/10.25368/2022.120.

[Baa+03] Franz Baader et al., eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BBL05] F. Baader, S. Brandt, and C. Lutz. "Pushing the $\mathcal{EL}$ Envelope". In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*. Edinburgh, UK: Morgan-Kaufmann Publishers, 2005.

[BM10] Franz Baader and Barbara Morawska. "Unification in the Description Logic $\mathcal{EL}$". In: *Logical Methods in Computer Science* 6.3 (2010). Special Issue of the 20th International Conference on Rewriting Techniques and Applications; also available at http://arxiv.org/abs/1006.2289.

[LW10] Carsten Lutz and Frank Wolter. "Deciding inseparability and conservative extensions in the description logic EL". In: *Journal of Symbolic Computation* 45.2 (2010), pp. 194–228.

[Baa+11a] Franz Baader et al. *Unification in the Description Logic $\mathcal{EL}$ Without the Top Concept*. LTCS-Report 11-01. See http://lat.inf.tu-dresden.de/research/reports.html. Dresden,

Germany: Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, 2011. DOI: https://doi.org/10.25368/2022.179.

[Baa+11b] Franz Baader et al. "Unification in the Description Logic $\mathcal{EL}$ without the Top Concept". In: *Proceedings of the 23rd International Conference on Automated Deduction (CADE 2011)*. Ed. by Nikolaj Bjørner and Viorica Sofronie-Stokkermans. Vol. 6803. Lecture Notes in Computer Science. Wroclaw, Poland: Springer-Verlag, 2011, pp. 70–84.

[BBM12] Franz Baader, Stefan Borgwardt, and Barbara Morawska. "Extending Unification in $\mathcal{EL}$ Towards General TBoxes". In: *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*. Ed. by Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith. AAAI Press, 2012, pp. 568–572.

[Baa+12] Franz Baader et al. "UEL: Unification Solver for $\mathcal{EL}$". In: *Proceedings of the 25th International Workshop on Description Logics (DL'12)*. Ed. by Yevgeny Kazakov, Domenico Lembo, and Frank Wolter. Vol. 846. CEUR Workshop Proceedings. Rome, Italy, 2012, pp. 26–36.

[ZT13a] Benjamin Zarrieß and Anni-Yasmin Turhan. *Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes*. LTCS-Report 13-06. See http://lat.inf.tu-dresden.de/research/reports.html. Dresden, Germany: Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, 2013. DOI: https://doi.org/10.25368/2022.196.

[ZT13b] Benjamin Zarrieß and Anni-Yasmin Turhan. "Most Specific Generalizations w.r.t. General $\mathcal{EL}$-TBoxes". In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. Beijing, China: AAAI Press, 2013.

[BM14a] Franz Baader and Barbara Morawska. "Matching with respect to general concept inclusions in the Description Logic $\mathcal{EL}$". In: *Proceedings of the 37th German Conference on Artificial Intelligence (KI'14)*. Ed. by Carsten Lutz and Michael Thielscher. Vol. 8736. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2014, pp. 135–146.

[BM14b] Franz Baader and Barbara Morawska. *Matching with respect to general concept inclusions in the Description Logic $\mathcal{EL}$*. LTCS-Report 14-03. Dresden, Germany: Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, 2014. DOI: https://doi.org/10.25368/2022.205.

[BF16] Franz Baader and Oliver Fernández Gil. *Extending the Description Logic $\tau\mathcal{EL}(deg)$ with Acyclic TBoxes*. LTCS-Report 16-02. See http://lat.inf.tu-dresden.de/research/reports.html. Dresden, Germany: Chair for Automata Theory, Institute for Theoretical Computer Science, Technische Universität Dresden, 2016. DOI: https://doi.org/10.25368/2022.226.