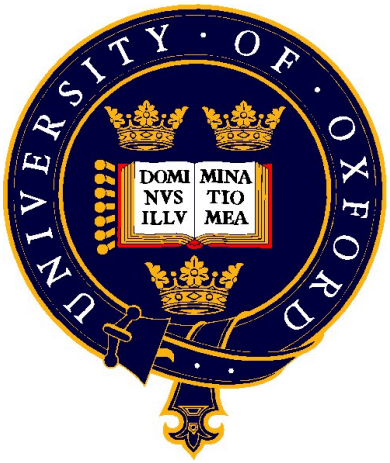
University of Oxford



Consequence-based Datatype Reasoning in $\mathcal{EL}$: Identifying the Tractable Fragments

by

Despoina Magka

St. Peter's College

under joint supervision by Prof. Ian Horrocks and Dr. Yevgeny Kazakov

A dissertation submitted in partial fulfilment of the degree of Master of Science in Computer Science.

Oxford University Computing Laboratory

September 2009

Candidate: Despoina Magka

Signed:..................................

Date:....................................

Abstract

The current dissertation suggests a consequence-based reasoning approach for the $\mathcal{EL}^{\perp}$ description logic with numerical datatypes. It provides a set of saturation rules which are used in the formulation of a polynomial classfication algorithm. Furthermore, it introduces the notion of safety which is a property that numerical datatypes exhibit and guarantees the preservation of polynomiality. The proposed algorithm is proved to be complete only for the safe datatypes. Additionally, the corresponding reasoning problem for non-safe datatypes is proved to be EXPTIME-hard. Apart from that, a classification of specific instances of datatypes is attempted. Finally, the present work, based on the results it produces, suggest a modification of the EL Profile in OWL 2 in order to add datatype features, which are currently available only in OWL 2.

Acknowledgements

First of all, I would like to express my most sincere thanks to my supervisors Yevgeny Kazakov and Ian Horrocks. Yevgeny patiently guided me for the last four months through the world of DL and constantly provided me without insightful remarks. Ian supplied me with constructive and invaluable comments. I would also like to thank Birte Glimm for her advice to datatype issues and the members of the Knowledge Representation and Reasoning group for creating an enjoyable working atmosphere.

Many thanks go to my parents, Giannis and Katerina, for their faithful and continuous support as well as to Giorgos for the encouraging and higly motivating skype calls; also to all my Oxford friends who made this year an unforgettable one.

Last but not least, I would like to acknowledge the substantial subsidy I received from the "John S. Latsis Ilean's Scholarship Foundation", without the aid of which the current project would not have been completed.

Contents

Chapter 1

Introduction

1.1 Description Logics

Artificial Intelligence, amongst other things, aims at bridging the gap between humans and machines. In the same spirit, Knowledge Representation and Reasoning, which is a branch of Artificial Intelligence, aspires to render human knowledge accessible to both humans and automated agents. Ontologies are a promising approach towards that direction. An ontology can be perceived as a formally defined set of entities and established relations between them. Ontologies have been heavily used to model domain knowledge in fields such as biology, medicine, geology, astronomy and defence. Another important application area of ontologies is the Semantic Web [10], where ontologies are used for information integration and annotation of web content [4]. OWL (Web Ontology Language)[1] is a W3C ontology language standard initially developed for use in the semantic web and now widely used both academically and commercially.

Description logics [3] (DL) are a family of logic-based languages that provide a firm mathematical background for the formulation and manipulation of ontologies. The DL syntax mainly employs *concepts*, *roles* and *individuals*. Individuals are similar to constants—they are names for elements of the domain. Concepts, like unary predicates, describe common characteristics shared by a set of individuals. Roles, in the same way as binary predicates, describe links between pairs of individuals. The constructors allow concepts and roles to be combined in order to describe new concepts. E.g., if we want to model in DL the concept of "a man who is a lecturer and has a young doctoral student", we may write:

$$\mathsf{Man} \sqcap \mathsf{Lecturer} \sqcap \exists \mathsf{hasDoctoralStudent}.\mathsf{Young} \tag{1.1}$$

where Man, Lecturer and Young are concepts and hasDoctoralStudent is a role. The constructor *conjunction* ($\sqcap$) is used to identify individuals that

[1] `http://www.w3.org/2004/OWL/`

are instances of each of the conjoined concepts and the *existential restriction* ($\exists r.C$) to identify individuals which are connected via the role r to an individual that is an instance of the concept C. DL is further equipped with a *terminological formalism* and an an *assertional formalism*. The terminological formalism includes axioms that allow the formulation of constraints such as:

$$\mathsf{HappyLecturer} \equiv \mathsf{Man} \sqcap \mathsf{Lecturer} \sqcap \exists \mathsf{hasDoctoralStudent}.\mathsf{Young} \tag{1.2}$$

$$\exists \mathsf{teaches}.\top \sqsubseteq \mathsf{Lecturer} \tag{1.3}$$

(1.3) uses $\top$ (*top* concept) which is the most general concept in the sense that all individuals are instances of it. (1.3) says that if an individual Ind teaches something (whatever it may be) then Ind is necessarily a lecturer. In DLs, a set of expressions of this form is called a TBox, but in applications it is usually referred to as an ontology; we will from now on refer to such a set of axioms as an ontology.

The *assertional formalism*, on the other hand, allows facts to be stated about individuals. E.g., we can introduce the individuals ALICE and BOB and make assertions such as Man(BOB) or hasDoctoralStudent(BOB,ALICE). A set of similar assertions comprises what we call an *ABox*. However, as the current dissertation focuses on the terminological rather than the assertional part of DLs, we will not often mention individuals from now on.

DLs have a well defined semantics based on a first order style model theory. Specifically, individuals are interpreted as set elements, concepts are interpreted as sets of these elements and roles interpreted as binary relations between the set elements. Constructors are interpreted in a same set-theoretical way. For example, the *conjunction* ($\sqcap$) is interpreted as an intersection between sets and the *disjunction* ($\sqcup$) as a union of them. As far as axioms are concerned (e.g., $A \sqsubseteq B$), the interpretation set of the subsumee concept (A) is always subset of or equal to the interpretation set of the subsumer concept (B). A set of axioms can have several different interpretations (*models*). An ontology entails a statement if it is the case that the statement is true in every model of the ontology.

Datatypes are similar to concepts, and represent sets of data values; the datatypes integer and string, for example, represent the sets of integer and string values respectively. Data values are similar to individuals and are instances of some datatype; the integer 25 and the string "Hello World" are, for example, values of the integer and string datatypes correspondingly. Features are similar to roles, and describe relationships between individuals and data values. Features are used in restrictions such as $\exists f.d$, which identifies those individuals that are linked via the feature f to a value of the datatype d. Numerical datatypes are particularly important, e.g., in

biomedical ontologies where they are needed to model information such as height, temperature, age and drug dosage.

The main difference between concepts and datatypes is that datatypes have a fixed or "concrete" interpretation that captures the structure of the relevant data values, and are usually equipped with a set of predicates that allow this structure to be examined. For example, an integer datatype may be equipped with comparison operators such as $<$ (less than), $=$ (equal to) and $>$ (greater than). These can be used to form a new datatype by restricting the range of values in one of the base datatypes. For example, $(<, 28)$ is a datatype whose interpretation consists of those integers less than 28. By using such a datatype in a restriction it is possible, for example, to assert that individuals whose age is less than 28 are instances of the concept Young:

$$\exists \mathsf{hasAge}.(<, 28) \sqsubseteq \mathsf{Young} \tag{1.4}$$

One can perform several reasoning tasks w.r.t. an ontology, such as checking whether a subsumption axiom holds; that is for $A \sqsubseteq B$ checking whether the ontology entails that the interpretation set of A is a subset of or equal to the interpretation set of B. It is often useful to compute the *concept hierarchy* (the subsumption quasi-ordering on concept names occurring in the ontology), a procedure known as classification.

Adding expressivity to a DL language can lead to increased reasoning complexity. As a consequence, the trade-off between expressivity and computational cost is a research issue of fundamental importance for the DL community. Finding a balance between reasoning complexity and adequate expressivity has been a field of thorough investigation and on each occasion the appropriate DL language is higly dependent on the application case. Identifying DLs whose key reasoning problems can be solved in polynomial time has been an important goal in DL research [11, 14, 6]. From now on we refer to such DLs as tractable, and to those that don't have this property as intractable.

DL datatype reasoning has also attracted a significant research interest. Several investigations have systematically explored the complexity and decidability of DL datatypes [8, 12]. Other works have suggested a specific datatype implementation for OWL, such as the use of an external datatype checker, which implies the separation of datatype reasoning from the remaining reasoning process [13].

1.2 Problem Statement

OWL is a highly expressive ontology language but reasoning in OWL, as a result of the abovementioned trade-off, is in 2NEXPTIME. Therefore, concerns over tractability have raised an increasing interest in restricted lan-

guages, with the new proposed version of OWL including language subsets that correspond to tractable DLs. Reasoners for expressive languages are typically based on tableau (model construction) algorithms, whereas reasoners for tractable languages typically use consequence-based techniques. Consequence-based reasoning can be thought as a proof construction, where a set of inference rules is used to add new axioms based on the existing axioms of the ontology.

The consequence-driven technique was first employed for the $\mathcal{EL}$ language, which includes top concept ($\top$), conjunction ($\sqcap$), existential restriction ($\exists r.C$) and some further expressivity. Using this technique it proved possible to develop a polynomial time classification algorithm for $\mathcal{EL}$, and a system based on this algorithm was able to classify the SNOMED CT medical terminology ontology, which describes more than 400,000 named concepts [7]. Later, the above $\mathcal{EL}$ language was extended to $\mathcal{EL}^{++}$ with polynomiality preserved. $\mathcal{EL}^{++}$ further included the bottom concept ($\bot$), nominals (singleton concepts), concrete domains (datatypes) and additional role features [1]. The encouraging complexity results of $\mathcal{EL}$ also led to the creation of the EL Profile of OWL 2[2], which is the corresponding language subset of OWL 2[3]. The culmination of this consequence-based investigation was the devising of a reasoning algorithm for the Horn $\mathcal{SHIQ}$ DL language, which corresponds to an intractable fragment of OWL. An implementation of that algorithm in the CB[4] was used to classify the largest available OWL version of GALEN, a biomedical ontology that had never been classified before (CB reasoner for Horn $\mathcal{SHIF}$) [9].

Given the importance of datatypes in biomedical applications and the increasing popularity of $\mathcal{EL}$, a suggestion has been made in order to incorporate datatypes (concrete domains) in $\mathcal{EL}$ [1]. That work has proposed a strict but coarse-grained framework on how to reason in $\mathcal{EL}$ in the presence of datatypes and without losing polynomiality. Nevertheless, no detailed description is given on how datatypes can be accommodated in the reasoning process. Moreover, it is not clear enough when numerical datatypes cause intractability and which are the properties that make them exhibit exponential complexity. Once boundaries w.r.t. retaining polynomiality are identified and the relevant issues are resolved the numerical datatypes will be integrated successfully to the consequence-based version of $\mathcal{EL}$ language.

1.3 Purpose and Structure of the Dissertation

The current dissertation mainly aims at studying how the numerical datatypes with operators ($<$, $>$, $\leq$, $\geq$, $=$) behave in conjunction with $\mathcal{EL}$ and

[2] `http://www.w3.org/TR/owl2-profiles/#OWL_2_EL`

[3] `http://www.w3.org/TR/owl2-syntax/`

[4] `http://code.google.com/p/cb-reasoner/reasoner`

consequence-based reasoning. Towards this end we introduce the notion of *safety* for *numerical datatypes with operators (NDOs)*, that guarantees the preservation of polynomiality. We propose a set of inference rules that accommodate datatype restrictions and formulate a consequence-based algorithm based on them. The algorithm consists of two stages: the normalization stage which performs a structural transformation on the input ontology and the saturation stage which exhaustively applies the inference rules to the transformed ontology. We prove soundness and completeness of the algorithm for the safe cases and show how the non-safe cases can cause intractability. Moreover, we attempt a classification of specific instances of numerical datatypes and suggest how the EL Profile of OWL 2 (W3C proposed recommendation) can be modified to allow more comprehensive use of numerical datatypes without losing tractability.

We now give a brief description of the dissertation structure:

- Chapter 2 provides the greatest part of the preliminary definitions which will later be used. It specifies the syntax and the semantics of the DLs to be examined as well as some reasoning tasks.
- Chapter 3 describes the normalization stage which is the first stage of the reasoning process we propose. We suggest a formal algorithm and prove termination, polynomiality and correctness.
- Chapter 4 deals with the saturation stage of the reasoning process and explores the complexity of reasoning. We define what NDOs are and when they are considered to be safe. We propose a sound and complete algorithm for safe NDOs and prove how non-safe NDOs lead to intractability.
- Chapter 5 extensively lists specific instances of safe and non-safe NDOs and offers a tractability guide to them.
- Chapter 6 puts forward a polynomial extension that can be made to the EL Profile of OWL 2 concerning additional use of numerical datatypes.
- Chapter 7 summarises the results, draws conclusions and indicates some future prospects.

Chapter 2

Preliminaries

The current chapter introduces the DL language to be studied and most of the necessary background notation. Section 2.1 formally defines the syntax of the language, Section 2.2 continues with the semantics and Section 2.3 indicates three reasoning tasks which are closely related to the algorithm described in the subsequent chapters.

2.1 Syntax

$\mathcal{EL}^{\bot}(D)$ is the $\mathcal{EL}^{\bot}$ Language [1] further extended with datatype restrictions with relational operators, such as $<$, $\leq$, $>$, $\geq$ and $=$. The syntax of $\mathcal{EL}^{\bot}(D)$ uses a set of *concept names* N_C, a set of *role names* N_R and a set of *feature names* N_F. $\mathcal{EL}^{\bot}(D)$ is parametrised with a *numerical datatype* D, such that $D \subseteq \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. N_C, N_R and N_F are countably infinite sets and, additionally, pairwise disjoint. The set of concepts is recursively defined using the constructors of Table 2.1, where C and E are concepts, $r \in N_R$, $F \in N_F$, $op \in \{<, \leq, >, \geq, =\}$ and $q \in D$. We call (op, q), where $op \in \{<, \leq, >, \geq, =\}$ and $q \in D$, a *D-datatype restriction* (or simply a *datatype restriction* if the datatype D is clear from the context). The expression v *op* q (Table 2.1), where $op \in \{<, \leq, >, \geq, =\}$, is interpreted as the corresponding standard relation on real numbers. We say that an $x \in D$ *satisfies* (op, q), iff x *op* q holds. We typically use the capital letters A, B to refer to concept names and the capital letters C, D or E to refer to concepts. We also set the abbreviation $N_C^{\top} = N_C \cup \{\top\}$.

An *axiom* α *in* $\mathcal{EL}^{\bot}(D)$ or simply *an axiom* α is either an expression of the form $C \sqsubseteq D$ or $\mathsf{Funct}(F)$, where C, D are concepts and $F \in N_F$. An *ontology* $\mathcal{O}$ *in* $\mathcal{EL}^{\bot}(D)$ or simply *an ontology* $\mathcal{O}$ is a set of axioms. We say that a concept E occurs in a concept C iff E is used as a concept in the construction of C. Moreover, a concept E is said to *positively (negatively) occur* in an axiom $C \sqsubseteq D$ iff it occurs in the concept D (C); we alternatively say that we have a *positive (negative) occurrence* of E.

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap E$	$C^{\mathcal{I}} \cap E^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
existential datatype restriction	$\exists F.(op, q)$	$\{x \in \Delta^{\mathcal{I}} \mid \exists v \in D : (x,v) \in F^{\mathcal{I}} \wedge (v\ op\ q)\}$

Table 2.1: Concept descriptions in $\mathcal{EL}^{\bot}(D)$.

Name	Syntax	Semantics
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
feature functionality	$\mathsf{Funct}(F)$	$(x,v_1) \in F^{\mathcal{I}} \wedge (x,v_2) \in F^{\mathcal{I}} \Rightarrow v_1 = v_2$

Table 2.2: Axioms in $\mathcal{EL}^{\bot}(D)$.

Example 1. *At this point, we introduce the ontology $\mathcal{O}_{ex}$, which we also use in the next chapters in terms of a running example. We set $D = \{r \in \mathbb{Z} \mid 0 \leq r \leq 120\}$. $\mathcal{O}_{ex}$ consists of the axioms α_1-α_5:*

α_1 $\mathsf{Child} \sqsubseteq \exists \mathsf{hasAge}.(<,12) \sqcap \exists \mathsf{hasAge}.(>,5)$

α_2 $\exists \mathsf{hasAge}.(<,18) \sqsubseteq \mathsf{MinorDosageAllowed}$

α_3 $\exists \mathsf{hasSymptom.Fever} \sqcap \mathsf{MinorDosageAllowed} \sqsubseteq \mathsf{MinorDosagePrescribed}$

α_4 $\mathsf{FeverishChild} \sqsubseteq \mathsf{Child} \sqcap \mathsf{Feverish}$

α_5 $\mathsf{Feverish} \sqsubseteq \exists \mathsf{hasSymptom.Fever}$

In the above example, Child, MinorDosageAllowed *and* Fever *are examples of concept names,* hasSymptom *of a role name,* hasAge *of a feature name and* $(>,5)$ *of a D-datatype restriction. Moreover, the concept* $\exists \mathsf{hasAge}.(<,18)$ *negatively occurs in α_2 and the concept* $\mathsf{Child} \sqcap \mathsf{Feverish}$ *(as well as* Child *and* Feverish*) positively occurs in α_4 .*

2.2 Semantics

An interpretation of $\mathcal{EL}^{\bot}(D)$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set which we call the *domain of the interpretation* and $\cdot^{\mathcal{I}}$ is the *interpretation function.* The interpretation function maps each concept name A to a subset

$A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each feature name $F \in N_F$ to a relation $F^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times D$. The constructors of $\mathcal{EL}^{\bot}(D)$ are interpretated as indicated in Table 2.1. For an axiom α we write $\mathcal{I} \models \alpha$ and we say that *an interpretation $\mathcal{I}$ satisfies an axiom α*, iff the corresponding semantics condition of Table 2.2 holds. If $\mathcal{I} \models \alpha$ for every $\alpha \in \mathcal{O}$ then $\mathcal{I}$ *is a model of* $\mathcal{O}$ and we write $\mathcal{I} \models \mathcal{O}$. Additionally, if every model $\mathcal{I}$ of $\mathcal{O}$ satisfies the axiom α then we say that $\mathcal{O}$ *entails* α and we write $\mathcal{O} \models \alpha$. We define the *signature of an ontology* $\mathcal{O}$ as the set $sig(\mathcal{O})$ of concept, role and feature names that occur in $\mathcal{O}$. Say that a concept, role or feature name X is *fresh w.r.t. an ontology* $\mathcal{O}$ iff $X \notin sig(\mathcal{O})$. If the ontology $\mathcal{O}$ is clear from the context then we just say that X is fresh. Given two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ and a set $S \subseteq N_C \uplus N_R \uplus N_F$, we write $\mathcal{I}|_S = \mathcal{J}|_S$ (and we say that *$\mathcal{I}$ and $\mathcal{J}$ coincide for the set S*) iff $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and $X^{\mathcal{I}} = X^{\mathcal{J}}$, for every $X \in S$.

2.3 Reasoning Tasks

Some of the most common reasoning tasks w.r.t. an ontology $\mathcal{O}$ are:

- Satisfiability: Checking whether a concept name A is satisfiable, that is whether $A^{\mathcal{I}}$ is non-empty for every model $\mathcal{I}$ of $\mathcal{O}$.
- Subsumption: Checking whether a concept name A is subsumed by a concept name B w.r.t. an ontology $\mathcal{O}$ ($\mathcal{O} \models A \sqsubseteq B$), that is if $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of the ontology $\mathcal{O}$.
- Classification: Classification of an ontology $\mathcal{O}$, that is computing all axioms of the form $A \sqsubseteq B$, where $A \in N_C^{\top} \cup \bot$ and $B \in N_C^{\top} \cup \bot$ are concept names and $\mathcal{O} \models A \sqsubseteq B$. The set of these axioms is called the *taxonomy* of the ontology $\mathcal{O}$.

In the following chapters, we will describe a procedure which performs the classification of an ontology $\mathcal{O}$. Specifically, the algorithm computes a set of axioms from which the taxonomy can easily be extracted. Checking satisfiability of a concept name A (or subsumption of an axiom $A \sqsubseteq B$) can be reduced to this process by verifying whether $A \sqsubseteq \bot$ (or $A \sqsubseteq B$) belongs to the computed taxonomy. The procedure consists of two stages: the normalization stage (which is described in Chapter 3) and the saturation stage (which is described in Chapter 4).

Chapter 3

Normalization Stage

Given an ontology $\mathcal{O}$, the normalization stage performs a structural transformation to it and produces the ontology $\mathcal{O}'$, which is in appropriate form to be used as an input for the saturation stage. In Section 3.1 we present the normal form in which axioms of $\mathcal{O}'$ should be and in Section 3.2 we describe a set of rules that are able to convert axioms in arbitrary form to axioms in normal form as well as a polynomial algorithm, which performs the above transformation.

3.1 Normal Forms of Axioms

An axiom is in *normal form* if it has one of the forms NF1-NF8 of Table 3.1, where $A, A_1, A_2, B \in N_C^\top$, $r \in N_R$, $F \in N_F$, $op \in \{<, \leq, >, \geq, =\}$ and $q \in D$ [1].

NF1	$A \sqsubseteq B$	NF2	$A_1 \sqcap A_2 \sqsubseteq B$
NF3	$A \sqsubseteq \exists r.B$	NF4	$\exists r.B \sqsubseteq A$
NF5	$A \sqsubseteq \exists F.(op, q)$	NF6	$\exists F.(op, q) \sqsubseteq A$
NF7	$A \sqsubseteq \bot$	NF8	$\mathsf{Funct}(F)$

Table 3.1: Axioms in normal form

3.2 Rules and Normalization Algorithm

In this section, we present a set of rules which can be used for the normalization of axioms. Specifically, if rules NR1-NR6 from Table 3.2 are exhaustively applied to axioms in arbitrary form of an ontology $\mathcal{O}$, they produce an ontology $\mathcal{O}'$ consisting of axioms in normal form [1]. For the rules NR1-NR6 of Table 3.2, we have that $B \in N_C^\top$, G, $H \notin N_C^\top$, C, D, E, G and H are concepts and NR1 is applied modulo commutativity of conjunction. The concept name A introduced at the conclusions of the right column is fresh

w.r.t. the ontology which has, so far, been transformed. This will become clearer as we formally define the algorithm.

NR	head	body
NR1	$C \sqcap H \sqsubseteq E$	$\{H \sqsubseteq A, C \sqcap A \sqsubseteq E\}$
NR2	$\exists R.G \sqsubseteq D$	$\{G \sqsubseteq A, \exists R.A \sqsubseteq D\}$
NR3	$G \sqsubseteq H$	$\{G \sqsubseteq A, A \sqsubseteq H\}$
NR4	$C \sqsubseteq \exists R.H$	$\{C \sqsubseteq \exists R.A, A \sqsubseteq H\}$
NR5	$B \sqsubseteq C \sqcap D$	$\{B \sqsubseteq C, B \sqsubseteq D\}$
NR6	$\bot \sqsubseteq C$	$\emptyset$

Table 3.2: The normalization rules for $\mathcal{EL}^{\bot}(D)$

We now provide a few definitions which are necessary for the formulation of the normalization algorithm.

Definition 1. *We define:*

- *The set N_{CC} as the set of all concepts*
- *The set Ax as the set of all axioms of the form $C \sqsubseteq D$ where C, $D \in N_{CC}$*

We say that a rule $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$ is applicable *to an axiom α if α matches the head of R. Let $\mathcal{O}$ be an ontology, $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$ and $\alpha \in Ax$. The function $\mathsf{R}(\alpha, \mathcal{O})$ returns the body of R (appropriately modified according to the matching of α with the head of R) if R is applicable to α, with the concept name A introduced to the body of R being fresh w.r.t. $\mathcal{O}$; otherwise, it returns the empty set.*

Example 2. *Let $\mathcal{O} = \{\exists r.(A \sqcap B) \sqsubseteq C \sqcap D, E \sqcap F \sqsubseteq \exists R.G\}$ and $\alpha = \exists r.(A \sqcap B) \sqsubseteq C \sqcap D$. In that case, we have $\mathsf{NR2}(\alpha, \mathcal{O}) = \{A \sqcap B \sqsubseteq H, \exists r.H \sqsubseteq C \sqcap D\}$.*

Algorithm 1, which is based on rules NR1-NR6, performs the tranformation we describe, i.e., the normalization of an ontology $\mathcal{O}$.

Example 3. *In order to clarify Algorithm 1 we apply it to $\mathcal{O}_{ex}$, which was introduced in Example 1 (page 8). We set $\mathcal{O} = \mathcal{O}_{ex}$. We do not describe in full detail the execution of the algorithm. The loop iterates 11 times: 3 times when $\mathsf{NR5}(\alpha_1, \mathcal{O}) \neq \emptyset$, $\mathsf{NR1}(\alpha_3, \mathcal{O}) \neq \emptyset$ and $\mathsf{NR5}(\alpha_4, \mathcal{O}) \neq \emptyset$ (line 5) and 8 times when axioms are moved from $\mathcal{O}$ to $\mathcal{O}'$ (lines 7-8). The output $\mathcal{O}'$ consists of the following axioms:*

Algorithm 1 : Normalization Algorithm

Input: = ontology $\mathcal{O}$ consisting of axioms in arbitrary form
Output: = ontology $\mathcal{O}'$ consisting of axioms in normal form

1: $\mathcal{O}' := \emptyset$;
2: **repeat**
3: choose an axiom $\alpha \in \mathcal{O}$;
4: **if** $\mathsf{R}(\alpha, \mathcal{O}) \neq \emptyset$ for some $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$ **then**
5: $\mathcal{O} := (\mathcal{O} \setminus \alpha) \cup \mathsf{R}(\alpha, \mathcal{O} \cup \mathcal{O}')$;
6: **else**
7: $\mathcal{O}' := \mathcal{O}' \cup \alpha$;
8: $\mathcal{O} := \mathcal{O} \setminus \alpha$;
9: **end if**
10: **until** $\mathcal{O} := \emptyset$;

α_{1a} $\mathsf{Child} \sqsubseteq \exists \mathsf{hasAge}.(<, 12)$

α_{1b} $\mathsf{Child} \sqsubseteq \exists \mathsf{hasAge}.(>, 5)$

α_2 $\exists \mathsf{hasAge}.(<, 18) \sqsubseteq \mathsf{MinorDosageAllowed}$

α_{3a} $\exists \mathsf{hasSymptom}.\mathsf{Fever} \sqsubseteq \mathsf{A}$

α_{3b} $\mathsf{A} \sqcap \mathsf{MinorDosageAllowed} \sqsubseteq \mathsf{MinorDosagePrescribed}$

α_{4a} $\mathsf{FeverishChild} \sqsubseteq \mathsf{Child}$

α_{4b} $\mathsf{FeverishChild} \sqsubseteq \mathsf{Feverish}$

α_5 $\mathsf{Feverish} \sqsubseteq \exists \mathsf{hasSymptom}.\mathsf{Fever}$

The rules which have been applied are:

- $\mathsf{NR5}(\alpha_1, \mathcal{O}) = \{\alpha_{1a}, \alpha_{1b}\}$ *with* $\mathcal{O} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$
- $\mathsf{NR1}(\alpha_3, \mathcal{O}) = \{\alpha_{3a}, \alpha_{3b}\}$ *with* $\mathcal{O} = \{\alpha_{1a}, \alpha_{1b}, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$
- $\mathsf{NR5}(\alpha_4, \mathcal{O}) = \{\alpha_{4a}, \alpha_{4b}\}$ *with* $\mathcal{O} = \{\alpha_{1a}, \alpha_{1b}, \alpha_2, \alpha_{3a}, \alpha_{3b}, \alpha_4, \alpha_5\}$

Note that A *is a fresh concept name w.r.t. the ontology* $\mathcal{O} = \{\alpha_{1a}, \alpha_{1b}, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ *since it does not occur in any of its axioms.*

We now prove, using the following lemma, that Algorithm 1 produces only axioms in normal form.

Lemma 1. *If none of the rules* NR1-NR6 *is applicable to* α*, then* α *is in normal form.*

Proof. We prove it by showing that if α is not in the form NF1-NF8, then at least one of the rules NR1-NR6 is applicable to α. If $\alpha = \mathsf{Funct}(F)$ then α is in NF8. If $\alpha = C \sqsubseteq D$, then we have that $C \notin N_C^\top$ or $D \notin N_C^\top$; otherwise, α would be in NF1. We distinguish cases:

- If $C = \bot$, then rule NR6 is applicable.
- If $D = \bot$, then α is in NF7.
- If $C \notin N_C^\top$ and $D \in N_C^\top$, then C has one of the following three forms:
  - $A_1 \sqcap A_2$, where $A_1 \notin N_C^\top$ or $A_2 \notin N_C^\top$, otherwise α would be in NF2. In this case, rule NR1 is applicable.
  - $\exists r.B$, where $B \notin N_C^\top$, otherwise α would be in NF4. In this case, rule NR2 is applicable.
  - $\exists F.(op, q)$. In this case α is in NF6.
- If $C \notin N_C^\top$ and $D \notin N_C^\top$, then rule NR3 is applicable.
- If $C \in N_C^\top$ and $D \notin N_C^\top$, then D has one of the following three forms:
  - $A_1 \sqcap A_2$. In this case, rule NR5 is applicable.
  - $\exists r.B$, where $B \notin N_C^\top$, otherwise α would be in NF3. In this case, rule NR4 is applicable.
  - $\exists F.(op, q)$. In this case α is in NF5.

□

Corollary 1. *The ontology* $\mathcal{O}'$ *as produced in Algorithm 1 contains only axioms in normal form.*

Proof. Algorithm 1 adds an axiom α to $\mathcal{O}'$ only at line 7 after checking that none of the rules NR1-NR6 is applicable to α. Since if none of the rules NR1-NR6 is applicable to α, then α is in normal form (from Lemma 1), only axioms in normal form are added to $\mathcal{O}'$. □

Our goal now is to prove termination of Algorithm 1. Intuitively, Algorithm 1 terminates since rules NR1-NR6 replace axioms with simpler ones. To formalize this idea, we use a notion of a multiset extension of well-founded orders. We firstly describe what a multiset and union of multisets is; we continue with the definition of a (well-founded) strict order and a multiset extension of a strict order to finally present Lemma 2; Lemma 2 is used in the termination proof.

Definition 2 (**Multiset and Operation on Multisets** [5])**.** *Let A be a set. We define a* multiset *M over A as a function $M : A \to \mathbb{N}$. The value $M(a)$ for $a \in A$ is called the* multiplicity *of a in M. A multiset is said to be* finite *if there are only finitely many $x \in A$ such that $M(x) > 0$. We say that an element x* belongs *to a multiset M ($x \in M$) iff $M(x) > 0$. The* union *$M \cup^{mul} N$ of two multisets $M : A \to \mathbb{N}$ and $N : A \to \mathbb{N}$ is defined by:*

$$(M \cup^{mul} N)(x) = M(x) + N(x)$$

Example 4. *Let $A = \mathbb{N}$. We define the multisets M and M' over $\mathbb{N}$ by $M(2) = 3$, $M(3) = 1$, $M'(2) = 1$, $M'(3) = 2$ and $M(n) = M'(n) = 0$ for $n \in \mathbb{N}$ and $n \neq 2, 3$. The abbreviations $M = \{2, 2, 2, 3\}$ and $M' = \{2, 3, 3\}$ are frequently used. The union between M and M' (using the abbreviation notation) is $M \cup^{mul} N = \{2, 2, 2, 2, 3, 3, 3\}$.*

Definition 3 (**Strictly ordered set** [5])**.** *A* strict order *on a set A is a binary relation $>$ (we use the infix notation for $>$), which is irreflexive (if $x \in A$, then $x \not> x$) and transitive (if $x > y$ and $y > z$, then $x > z$). A strictly ordered set is a pair $(A, >)$, where A is a set and $>$ a strict order on A.*

Definition 4 (**Multiset extension on strict order** [5])**.** *Let $(A, >)$ be a strictly ordered set. The* multiset extension *$>_{mul}$ of $>$ to finite multisets over A is defined by:*

$$S_1 >_{mul} S_2$$

$$\Leftrightarrow S_1 \neq S_2 \wedge$$

$$\forall m \in A : [S_2(m) > S_1(m) \Rightarrow \exists m' \in A : (m' > m \ \wedge \ S_1(m') > S_2(m'))]$$

Example 5. *Continuing Example 4 and given that $(\mathbb{N}, >)$ is a strict order, $M' >_{mul} M$ because $M(2) > M'(2)$ but $M'(3) > M(3)$.*

Definition 5 (**Well-founded strict order** [5])**.** *A strict order $(A, >)$ is said to be* well-founded *if there is no infinite decreasing chain such that:*

$$x_1 > x_2 > x_3 > \ldots$$

with $x_i \in A$.

Lemma 2 (**Well-founded Multiset extension** [5])**.** *A multiset extension $>_{mul}$ of a well-founded order, is well-founded as well.*

We say that X is a *symbol* if $X \in N_C \uplus N_R \uplus N_F$ and we define the size of an axiom α, $size : Ax \to \mathbb{N}$, as the number of symbols that α contains. Similarly, we define the size of a concept C, $size : N_{CC} \to \mathbb{N}$, as the number of symbols that C contains. In both cases, multiple occurrences count multiple times. Since $\top$ and $\bot$ are concepts that do not contain any symbols, we set $size(\top) = 1$ and $size(\bot) = 2$.

Lemma 3. *For* $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$*, an ontology* $\mathcal{O}$ *and every* α_1 *and* α_2 *such that* α_1 *matches rule* R *and* $\alpha_2 \in \mathsf{R}(\alpha, \mathcal{O})$*, we have that* $size(\alpha_1) > size(\alpha_2)$*.*

Proof. We prove the lemma by distinguishing six cases for the six different rules:

- NR1

  In this case $size(C \sqcap H \sqsubseteq E) = size(C) + size(H) + size(E)$ with $size(C) + size(H) + size(E) > size(H) + 1 = size(H \sqsubseteq A)$ and $size(C)+size(H)+size(E) > size(C)+1+size(E) = size(C \sqcap A \sqsubseteq E)$.

- NR2

  In this case $size(\exists R.G \sqsubseteq D) = 1+size(G)+size(D)$ with $1+size(G)+size(D) > size(G) + 1 = size(G \sqsubseteq A)$ and $1 + size(G) + size(D) > 1 + 1 + size(D) = size(\exists R.A \sqsubseteq D)$.

- NR3

  In this case $size(G \sqsubseteq H) = size(G)+size(H)$ with $size(G)+size(H) > size(G) + 1 = size(G \sqsubseteq A)$ and $size(G) + size(H) > 1 + size(H) = size(A \sqsubseteq H)$.

- NR4

  In this case $size(C \sqsubseteq \exists R.H) = size(C) + 1 + size(H)$ with $size(C) + 1 + size(H) > size(C) + 1 + 1 = size(C \sqsubseteq \exists R.A)$ and $size(C) + 1 + size(H) > 1 + size(H) = size(A \sqsubseteq H)$.

- NR5

  In this case $size(B \sqsubseteq C \sqcap D) = 1+size(C)+size(D)$ with $1+size(C)+size(D) > 1 + size(C) = size(B \sqsubseteq C)$ and $1 + size(C) + size(D) > 1 + size(D) = size(B \sqsubseteq D)$.

- NR6

  $size(\alpha_1) > size(\alpha_2)$ holds trivially because $size(\alpha_2) = 0$ and $size(\alpha_1) = size(\bot) + size(C) = 2 + size(C) > 0$.

□

Now, we are ready to continue with the termination proof of Algorithm 1.

Proposition 1 (**Termination of Normalization Algorithm**)**.** *Algorithm 1 terminates.*

Proof. Let Ax be a set of axioms. We define a strict order $>$ over Ax by $\alpha_1 > \alpha_2$ iff $size(\alpha_1) > size(\alpha_2)$, where $\alpha_1, \alpha_2 \in Ax$. The strict order $>$ over Ax is well-founded because there is no infinite descending chain such that $size(\alpha_1) > size(\alpha_2) > size(\alpha_3) > \ldots$ and $\alpha_i \in Ax$. We define the multiset S over Ax, $S : Ax \to \mathbb{N}$ and the mapping $multi$ from a set of axioms (an ontology) to a multiset S ($multi : \mathcal{P}(Ax) \to S$) by:

$$multi(\mathcal{O}) = \bigcup_{\alpha \in \mathcal{O}}^{mul} \{\alpha\}$$

We also define the multiset extension $>_{mul}$ of the well-founded strict order $>$ over Ax to S. $>_{mul}$ is well-founded as well from Lemma 2.

Claim 1. *Let $\mathcal{O}_i$ be the ontology $\mathcal{O}$ before and $\mathcal{O}_{i+1}$ be the ontology $\mathcal{O}$ after the execution of algorithm's loop. Then $multi(\mathcal{O}_i) >_{mul} multi(\mathcal{O}_{i+1})$.*

Proof. There are two possibilities for the algorithm's loop:

- $\mathsf{R}(\alpha, \mathcal{O}) \neq \emptyset$, for some $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$. In that case, $multi(\mathcal{O}_i) >_{mul} multi(\mathcal{O}_{i+1})$, because in $multi(\mathcal{O}_i)$, α is replaced with α_1 and α_2, where $\alpha_1, \alpha_2 \in \mathsf{R}(\alpha, \mathcal{O})$. However, from Lemma 3 $size(\alpha) > size(\alpha_1)$ and $size(\alpha) > size(\alpha_2)$ and, thus, $\alpha > \alpha_1$ and $\alpha > \alpha_2$. Since $multi(\mathcal{O}_i)(\alpha) > multi(\mathcal{O}_{i+1})(\alpha)$ and there is no α', such that $\alpha' > \alpha$ and $multi(\mathcal{O}_{i+1})(\alpha') > multi(\mathcal{O}_i)(\alpha)$, we have that $multi(\mathcal{O}_i) >_{mul} multi(\mathcal{O}_{i+1})$.
- $\mathsf{R}(\alpha, \mathcal{O}) = \emptyset$, for every $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$. In that case, $multi(\mathcal{O}_i) >_{mul} multi(\mathcal{O}_{i+1})$, because we remove an axiom α from the ontology $\mathcal{O}_i$ and, hence, the multiplicity of α in $multi(\mathcal{O}_{i+1})$ is reduced by one. Again, since $multi(\mathcal{O}_i)(\alpha) > multi(\mathcal{O}_{i+1})(\alpha)$ and there is no α', such that $\alpha' > \alpha$ and $multi(\mathcal{O}_{i+1})(\alpha') > multi(\mathcal{O}_i)(\alpha)$, we have that $multi(\mathcal{O}_i) >_{mul} multi(\mathcal{O}_{i+1})$.

□

Let $\mathcal{O}_0 = \mathcal{O}$ be the input of Algorithm 1; suppose that $\mathcal{O}_1$ is the ontology $\mathcal{O}$ after the first execution of the loop, $\mathcal{O}_2$ is the ontology $\mathcal{O}$ after the second execution and so on. From Claim 1 we have that $multi(\mathcal{O}_1) >_{mul} multi(\mathcal{O}_2) >_{mul} multi(\mathcal{O}_3) >_{mul} \ldots$. Since the defined multiset extension is well-founded, there is no infinite descending chain such that $multi(\mathcal{O}_1) >_{mul} multi(\mathcal{O}_2) >_{mul} multi(\mathcal{O}_3) >_{mul} \ldots$. Therefore, after a finite number of loop iterations there is $multi(\mathcal{O}_k)$, such that there exists no $\mathcal{O}_{k+1}$ with $multi(\mathcal{O}_k) > multi(\mathcal{O}_{k+1})$. However, this is true only when $multi(\mathcal{O}_k) = \emptyset$ which means that $\mathcal{O}_k = \emptyset$. $\mathcal{O}_k = \emptyset$ ($\mathcal{O} = \emptyset$) is the termination condition for Algorithm 1 and, thus, the algorithm terminates.

□

After proving that Algorithm 1 terminates we examine its complexity. The termination proof we presented did not provide us with any complexity bounds and, thus, we need to give a different argument concerning complexity. Algorithm 1 constitutes the first part of the reasoning process that the current dissertation describes and since we later intend to prove that the whole reasoning process is polynomial, it is necessary to prove that Algorithm 1 is at most polynomial.

Proposition 2 (**Complexity of Normalization Algorithm**). *Algorithm 1 runs in linear space and linear time.*

Proof. Suppose that $\mathcal{O} \cup \mathcal{O}'$ after the initialization of line 1 contains k_{all} constructors (that is occurrences of $\sqcap$, $\exists$ and $\bot$ in its axioms) and k_f axioms in NF8. We define a *positive conjunction* of an axiom as the positive occurrence of the constructor $\sqcap$. We denote the number of positive conjunctions of the axioms contained in the input ontology $\mathcal{O}$ with $k_\sqcap$ and the number of remaining constructors with k_{rcon}; thus, $k_{all} = k_\sqcap + k_{rcon}$.

Claim 2. *Algorithm 1 runs in linear space.*

We calculate the maximum number of axioms that Algorithm 1 can produce in the output. The resulting axioms can either be of the form NF1 (say k_1) or NF2-NF7 (say k_2) or NF8 (say k_3). An axiom in $\mathcal{O}'$ can be in NF1 only as a result of the application of NR5, since B (from the head of NR5 which is $B \sqsubseteq C \sqcap D$) is a concept name. Every application of NR5 results to the addition of at most two axioms in NF1. Thus, the maximum number for k_1 is $2k_\sqcap$. If the axiom is of the form NF2-NF7 then it contains exactly one constructor, which is not a positive conjunction. The initial ontology contains k_{rcon} such constructors and according to NR1-NR4, k_{rcon} is preserved. Thus, the maximum number for k_2 is k_{rcon}. Finally, the maximum number of axioms in NF8 is $k_3 = k_f$. In total, $\mathcal{O}'$ can contain at most $k_1 + k_2 + k_3 = 2k_\sqcap + k_{rcon} + k_f \leq 2k_{all} + k_f$ axioms, which is linear in the size of input $k_{all} + k_f$.

Claim 3. *Algorithm 1 runs in linear time.*

Concerning the time which is needed for the execution of the algorithm we calculate the number of loop iterations of Algorithm 1, that is how many times rules NR1-NR6 are applied. We observe that each axiom in NF1-NF7 has at most three occurrences of concept names. For rules NR1-NR4, the application of a rule is accompanied with the introduction of a fresh concept name. Thus, the maximum number of rule applications of NR1-NR4 can be found by multiplying the maximum number of axioms in $\mathcal{O}'$ and in NF1-NF7 (that is $2k_{all}$ as shown) with the maximum number of concept names that an axiom can contain (that is 3). Hence, rules NR1-NR4 can be executed no more than $6k_{all}$ times. NR5 can be applied at most $k_\sqcap$ times. Therefore, an

upper time bound for Algorithm 1 is $6k_{all} + k_{\sqcap} \leq 7k_{all}$ which is linear in the size of the input $k_{all} + k_f$. □

Now, we continue with two lemmas (Lemma 4 and Lemma 5) which are later on used in order to prove that Algorithm 1 is correct. Given that $\mathcal{O}$ is an ontology and $\mathcal{O}'$ the ontology we receive after the application of one of the $\{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$, Lemma 4 proves that all models of $\mathcal{O}'$ are models of $\mathcal{O}$ as well; Lemma 5 on the other hand shows that a model of $\mathcal{O}$ can always be extended to a model of $\mathcal{O}'$, too. Both lemmas are necessary in order to support the correctness proof of Algorithm 1, which intuitively says that an axiom (that contains no fresh concepts) is entailed by $\mathcal{O}$ iff it is entailed by $\mathcal{O}'$.

Lemma 4. *Let $\mathcal{O}$ be an ontology, $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$ and $\mathcal{O}' = (\mathcal{O} \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$. Then for every interpretation $\mathcal{I}$:*

$$\mathcal{I} \models \mathcal{O}' \Rightarrow \mathcal{I} \models \mathcal{O}$$

Proof. We assume that:

$$\mathcal{I} \models \mathcal{O}' \tag{3.1}$$

and

$$\mathcal{O}' = (\mathcal{O} \setminus \alpha) \cup \mathsf{R}(\alpha, \mathcal{O}) \tag{3.2}$$

and we need to prove that $\mathcal{I} \models \mathcal{O}$, that is $\mathcal{I} \models \alpha$, $\forall \alpha \in \mathcal{O}$. Take an arbitrary $\alpha \in \mathcal{O}$. If $\alpha \in \mathcal{O}'$, then from (3.1), $\mathcal{I} \models \alpha$. Otherwise from (3.2), α matches the head of R. We distinguish cases depending on the normalization rule which was applied:

NR1 We need to show $\mathcal{I} \models C \sqcap H \sqsubseteq E$ or $(C \sqcap H)^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. From (3.1) and $H \sqsubseteq A$, $C \sqcap A \sqsubseteq E \in \mathcal{O}'$, we derive that $\mathcal{I} \models H \sqsubseteq A$ and $\mathcal{I} \models C \sqcap A \sqsubseteq E$, which is $H^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ and $C^{\mathcal{I}} \cap A^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. From $H^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ we have that $C^{\mathcal{I}} \cap H^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap A^{\mathcal{I}}$. Using $C^{\mathcal{I}} \cap A^{\mathcal{I}} \subseteq E^{\mathcal{I}}$, we finally get $C^{\mathcal{I}} \cap H^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ or $(C \sqcap H)^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ which we wanted to prove.

NR2 We need to prove $\mathcal{I} \models \exists r.G \sqsubseteq D$ or $(\exists r.G)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. We need to prove that for every $x \in (\exists r.G)^{\mathcal{I}}$ we have $x \in D^{\mathcal{I}}$. From (3.1) and $G \sqsubseteq A$, $\exists r.A \sqsubseteq D \in \mathcal{O}'$, we derive that $\mathcal{I} \models G \sqsubseteq A$ and $\mathcal{I} \models \exists r.A \sqsubseteq D$, that is:

$$G^{\mathcal{I}} \subseteq A^{\mathcal{I}} \tag{3.3}$$

$$(\exists r.A)^{\mathcal{I}} \subseteq D^{\mathcal{I}} \tag{3.4}$$

Since $x \in (\exists r.G)^{\mathcal{I}}$, we have that there exists a y such that $(x, y) \in r^{\mathcal{I}}$ and $y \in G^{\mathcal{I}}$. From (3.3), $y \in A^{\mathcal{I}}$. Since there exists a y such that $(x, y) \in r^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}$, we have $x \in (\exists r.A)^{\mathcal{I}}$, and from (3.4), we have $x \in D^{\mathcal{I}}$, which we needed to prove.

NR3 We have to prove $\mathcal{I} \models G \sqsubseteq H$ or $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$. From (3.1), $G \sqsubseteq A \in \mathcal{O}'$ and $A \sqsubseteq H \in \mathcal{O}'$, we derive that $\mathcal{I} \models G \sqsubseteq A$ and $\mathcal{I} \models A \sqsubseteq H$, that is $G^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ and $A^{\mathcal{I}} \subseteq H^{\mathcal{I}}$. $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$ follows directly.

NR4 We have to show $\mathcal{I} \models C \sqsubseteq \exists r.H$ or $C^{\mathcal{I}} \subseteq (\exists r.H)^{\mathcal{I}}$. It is sufficient to take an x such that $x \in C^{\mathcal{I}}$ and prove that $x \in (\exists r.H)^{\mathcal{I}}$. From (3.1) and $C \sqsubseteq \exists r.A$, $A \sqsubseteq H \in \mathcal{O}'$, we derive that $\mathcal{I} \models C \sqsubseteq \exists r.A$ and $\mathcal{I} \models A \sqsubseteq H$, which is:

$$C^{\mathcal{I}} \subseteq (\exists r.A)^{\mathcal{I}} \tag{3.5}$$

$$A^{\mathcal{I}} \subseteq H^{\mathcal{I}} \tag{3.6}$$

From (3.5) and $x \in C^{\mathcal{I}}$, we have that $x \in (\exists r.A)^{\mathcal{I}}$, that is there exists a y such that $(x, y) \in r^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}$. Also, from (3.6) and $y \in A^{\mathcal{I}}$ we have that $y \in H^{\mathcal{I}}$ and, therefore, $x \in (\exists r.H)^{\mathcal{I}}$ which was required to show.

NR5 We need to show $\mathcal{I} \models B \sqsubseteq C \sqcap D$ or $B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap D^{\mathcal{I}}$. From (3.1) and $B \sqsubseteq C$, $B \sqsubseteq D \in \mathcal{O}'$, we derive that $\mathcal{I} \models B \sqsubseteq C$ and $\mathcal{I} \models B \sqsubseteq D$, which is $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. As a consequence, $B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap D^{\mathcal{I}}$ or $B^{\mathcal{I}} \subseteq (C \sqcap D)^{\mathcal{I}}$.

NR6 We need to prove $\mathcal{I} \models \bot \sqsubseteq C$ or $(\bot)^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, which is trivial since $(\bot)^{\mathcal{I}} = \emptyset$.

□

Lemma 5. *Let $\mathcal{O}$ be an ontology, $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$, $\mathcal{O}' = (\mathcal{O} \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$ and $\mathcal{I}$ a model such that $\mathcal{I} \models \mathcal{O}$. Then there exists an interpretation $\mathcal{J}$, such that $\mathcal{I}|_{sig(\mathcal{O})} = \mathcal{J}|_{sig(\mathcal{O})}$ and $\mathcal{J} \models \mathcal{O}'$.*

Proof. We assume that $\mathcal{I} \models \mathcal{O}$ and distinguish cases according to the normalization rule which was applied:

(NR1) We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ and:

$$X^{\mathcal{J}} := \begin{cases} H^{\mathcal{I}} \text{ if } X = A, \\ X^{\mathcal{I}} \text{ if } X \in N_C \uplus N_R \uplus N_F \text{ and } X \neq A. \end{cases} \tag{3.7}$$

We prove that $\mathcal{J} \models \mathcal{O}'$. Take an arbitrary $\alpha \in \mathcal{O}'$.

If $\alpha \in \mathcal{O}$, then from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \alpha$. The axiom α does not contain A, which is fresh w.r.t. $\mathcal{O}$ and since $\mathcal{I}|_{sig(\mathcal{O})} = \mathcal{J}|_{sig(\mathcal{O})}$, we have that $\mathcal{J} \models \alpha$.

If $\alpha = H \sqsubseteq A$, we need to prove that $\mathcal{J} \models H \sqsubseteq A$ and, thus, $H^{\mathcal{J}} \subseteq A^{\mathcal{J}}$. From (3.7), $H^{\mathcal{J}} = H^{\mathcal{I}}$ (since A does not occur in H) and $A^{\mathcal{J}} = H^{\mathcal{I}}$. $H^{\mathcal{J}} \subseteq A^{\mathcal{J}}$ follows trivially.

If $\alpha = C \sqcap A \sqsubseteq E$, it is sufficient to show that $\mathcal{J} \models C \sqcap A \sqsubseteq E$ that is $(C \sqcap A)^{\mathcal{J}} \subseteq E^{\mathcal{J}}$. Since NR1 was applied, we have that $C \sqcap H \sqsubseteq E \in \mathcal{O}$ and from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models C \sqcap H \sqsubseteq E$. Thus, $(C \sqcap H)^{\mathcal{I}} \subseteq E^{\mathcal{I}}$ or $C^{\mathcal{I}} \cap H^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. Using (3.7) and since A does not occur neither in C nor in E we have that $C^{\mathcal{I}} = C^{\mathcal{J}}$, $H^{\mathcal{I}} = A^{\mathcal{J}}$ and $E^{\mathcal{I}} = E^{\mathcal{J}}$. As a consequence, $C^{\mathcal{J}} \cap A^{\mathcal{J}} \subseteq E^{\mathcal{J}}$ or $(C \sqcap A)^{\mathcal{J}} \subseteq E^{\mathcal{J}}$, which we needed to prove.

(NR2) We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ and:

$$X^{\mathcal{J}} := \begin{cases} G^{\mathcal{I}} \text{ if } X = A, \\ X^{\mathcal{I}} \text{ if } X \in N_C \uplus N_R \uplus N_F \text{ and } X \neq A. \end{cases} \tag{3.8}$$

We prove that $\mathcal{J} \models \mathcal{O}'$. Take an arbitrary $\alpha \in \mathcal{O}'$.

If $\alpha \in \mathcal{O}$, then from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \alpha$. The axiom α does not contain A, which is fresh w.r.t. $\mathcal{O}$ and since $\mathcal{I}|_{sig(\mathcal{O})} = \mathcal{J}|_{sig(\mathcal{O})}$, we have that $\mathcal{J} \models \alpha$.

If $\alpha = G \sqsubseteq A$, we need to prove that $\mathcal{J} \models G \sqsubseteq A$ and, thus, $G^{\mathcal{J}} \subseteq A^{\mathcal{J}}$. From (3.8), $G^{\mathcal{J}} = G^{\mathcal{I}}$ (since A does not occur in G) and $A^{\mathcal{J}} = G^{\mathcal{I}}$. $G^{\mathcal{J}} \subseteq A^{\mathcal{J}}$ follows trivially.

If $\alpha = \exists r.A \sqsubseteq D$, it is sufficient to show that $\mathcal{J} \models \exists r.A \sqsubseteq D$ that is $(\exists r.A)^{\mathcal{J}} \subseteq D^{\mathcal{J}}$. Since NR2 was applied, we have that $\exists r.G \sqsubseteq D \in \mathcal{O}$ and from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \exists r.G \sqsubseteq D$. Thus, $(\exists r.G)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Using (3.8) and the interpretation of existential restriction we have that $(\exists r.G)^{\mathcal{I}} = \{x \mid (x,y) \in r^{\mathcal{I}} \wedge y \in G^{\mathcal{I}}\} = \{x \mid (x,y) \in r^{\mathcal{J}} \wedge y \in A^{\mathcal{J}}\} = (\exists r.A)^{\mathcal{J}}$. From (3.8), and since A does not occur in D we have $D^{\mathcal{I}} = D^{\mathcal{J}}$. Using $(\exists r.A)^{\mathcal{J}} = (\exists r.G)^{\mathcal{I}}$, $(\exists r.G)^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $D^{\mathcal{I}} = D^{\mathcal{J}}$ we derive $(\exists r.A)^{\mathcal{J}} \subseteq D^{\mathcal{J}}$.

(NR3) We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ and:

$$X^{\mathcal{J}} := \begin{cases} H^{\mathcal{I}} \text{ if } X = A, \\ X^{\mathcal{I}} \text{ if } X \in N_C \uplus N_R \uplus N_F \text{ and } X \neq A. \end{cases} \tag{3.9}$$

We prove that $\mathcal{J} \models \mathcal{O}'$. Take an arbitrary $\alpha \in \mathcal{O}'$.

If $\alpha \in \mathcal{O}$, then from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \alpha$. The axiom α does not contain A, which is fresh w.r.t. $\mathcal{O}$ and since $\mathcal{I}|_{sig(\mathcal{O})} = \mathcal{J}|_{sig(\mathcal{O})}$, we have that $\mathcal{J} \models \alpha$.

If $\alpha = G \sqsubseteq A$, we need to prove that $\mathcal{J} \models G \sqsubseteq A$ and, thus, $G^{\mathcal{J}} \subseteq A^{\mathcal{J}}$. From (3.9), $G^{\mathcal{J}} = G^{\mathcal{I}}$ (since A does not occur in G) and $A^{\mathcal{J}} = H^{\mathcal{I}}$. Thus, we need to show that $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$ which follows from $G \sqsubseteq H \in \mathcal{O}$ and $\mathcal{I} \models \mathcal{O}$.

If $\alpha = A \sqsubseteq H$, we need to prove that $\mathcal{J} \models A \sqsubseteq H$ and, thus, $A^{\mathcal{J}} \subseteq H^{\mathcal{J}}$. From (3.9), $A^{\mathcal{J}} = H^{\mathcal{I}}$ and $H^{\mathcal{J}} = H^{\mathcal{I}}$ (since A does not occur in H). Thus, $A^{\mathcal{J}} \subseteq H^{\mathcal{J}}$ follows trivially.

(NR4) We define an interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ such that $\Delta^{\mathcal{J}} := \Delta^{\mathcal{I}}$ and:

$$X^{\mathcal{J}} := \begin{cases} H^{\mathcal{I}} \text{ if } X = A, \\ X^{\mathcal{I}} \text{ if } X \in N_C \uplus N_R \uplus N_F \text{ and } X \neq A. \end{cases} \tag{3.10}$$

We prove that $\mathcal{J} \models \mathcal{O}'$. Take an arbitrary $\alpha \in \mathcal{O}'$.

If $\alpha \in \mathcal{O}$, then from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \alpha$. The axiom α does not contain A, which is fresh w.r.t. $\mathcal{O}$ and since $\mathcal{I}|_{sig(\mathcal{O})} = \mathcal{J}|_{sig(\mathcal{O})}$, we have that $\mathcal{J} \models \alpha$.

If $\alpha = C \sqsubseteq \exists r.A$, it is sufficient to show that $\mathcal{J} \models C \sqsubseteq \exists r.A$ that is $C^{\mathcal{J}} \subseteq (\exists r.A)^{\mathcal{J}}$. Since NR4 was applied, we have that $C \sqsubseteq \exists r.H \in \mathcal{O}$ and from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models C \sqsubseteq \exists r.H$. Thus, $C^{\mathcal{I}} \subseteq (\exists r.H)^{\mathcal{I}}$. Using (3.10) and the interpretation of existential restriction we have that $(\exists r.H)^{\mathcal{I}} = \{x \mid (x,y) \in r^{\mathcal{I}} \wedge y \in H^{\mathcal{I}}\} = \{x \mid (x,y) \in r^{\mathcal{J}} \wedge y \in A^{\mathcal{J}}\} = (\exists r.A)^{\mathcal{J}}$. From (3.10), and since A does not occur in C we have $C^{\mathcal{J}} = C^{\mathcal{I}}$. Using $C^{\mathcal{J}} = C^{\mathcal{I}}$, $C^{\mathcal{I}} \subseteq (\exists r.H)^{\mathcal{I}}$ and $(\exists r.H)^{\mathcal{I}} = (\exists r.A)^{\mathcal{J}}$ we derive $C^{\mathcal{J}} \subseteq (\exists r.A)^{\mathcal{J}}$.

If $\alpha = A \sqsubseteq H$, we need to prove that $\mathcal{J} \models A \sqsubseteq H$ and, thus, $A^{\mathcal{J}} \subseteq H^{\mathcal{J}}$. From (3.10), $H^{\mathcal{J}} = H^{\mathcal{I}}$ (since A does not occur in H) and $A^{\mathcal{J}} = H^{\mathcal{I}}$. $A^{\mathcal{J}} \subseteq H^{\mathcal{J}}$ follows trivially.

(NR5) We define the interpretation $\mathcal{J} = \mathcal{I}$ and we prove that $\mathcal{J} \models \mathcal{O}'$. Take an arbitrary $\alpha \in \mathcal{O}'$.

If $\alpha \in \mathcal{O}$, then from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models \alpha$. Since $\mathcal{J} = \mathcal{I}$, $\mathcal{J} \models \alpha$ as well.

If $\alpha = B \sqsubseteq C$, we need to prove that $\mathcal{J} \models B \sqsubseteq C$ and, thus, $B^{\mathcal{J}} \subseteq C^{\mathcal{J}}$. Since NR5 was applied, we have that $B \sqsubseteq C \sqcap D \in \mathcal{O}$ and from $\mathcal{I} \models \mathcal{O}$, $\mathcal{I} \models B \sqsubseteq C \sqcap D$. Thus, $B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap D^{\mathcal{I}}$. Since $\mathcal{J} = \mathcal{I}$, we have $B^{\mathcal{J}} \subseteq C^{\mathcal{J}} \cap D^{\mathcal{J}}$ and $B^{\mathcal{J}} \subseteq C^{\mathcal{J}}$ follows directly.

If $\alpha = B \sqsubseteq D$, we need to prove that $\mathcal{J} \models B \sqsubseteq D$ and, thus, $B^{\mathcal{J}} \subseteq D^{\mathcal{J}}$. As shown before, $B^{\mathcal{I}} \subseteq C^{\mathcal{I}} \cap D^{\mathcal{I}}$. Since $\mathcal{J} = \mathcal{I}$, we have $B^{\mathcal{J}} \subseteq C^{\mathcal{J}} \cap D^{\mathcal{J}}$ and $B^{\mathcal{J}} \subseteq D^{\mathcal{J}}$ follows directly.

(NR6) We define the interpretation $\mathcal{J} = \mathcal{I}$ and $\mathcal{J} \models \mathcal{O}'$ is trivial because no new axiom is added to $\mathcal{O}'$.

□

After proving Lemma 4 and Lemma 5, we continue to the correctness proof of Algorithm 1.

Theorem 6 (**Correctness of Normalization Algorithm**). *Let $\mathcal{O}$ be an ontology in $\mathcal{EL}^{\bot}(D)$ and $\mathcal{O}'$ the ontology obtained after applying Algorithm 1 to $\mathcal{O}$. For every axiom α, that does not contain any fresh concepts w.r.t. $\mathcal{O}$, (1) and (2) are equivalent:*

(1) $\mathcal{O} \models \alpha$

(2) $\mathcal{O}' \models \alpha$

Proof. Let $\mathcal{O}_0 = \mathcal{O} \cup \mathcal{O}'$ after the initialization (line 1) of Algorithm 1, $\mathcal{O}_1 = \mathcal{O} \cup \mathcal{O}'$ after the first loop iteration, $\mathcal{O}_2 = \mathcal{O} \cup \mathcal{O}'$ after the second and, so on. Let, also $\mathcal{O}_n = \mathcal{O} \cup \mathcal{O}'$ after the termination of the algorithm. Since $\mathcal{O}' = \emptyset$ at the begining and $\mathcal{O} = \emptyset$ at the end, we have that $\mathcal{O}_0 = \mathcal{O}$ and $\mathcal{O}_n = \mathcal{O}'$. Therefore, it suffices to show $\mathcal{O}_0 \models \alpha \Leftrightarrow \mathcal{O}_n \models \alpha$. We prove $\mathcal{O}_i \models \alpha \Leftrightarrow \mathcal{O}_{i+1} \models \alpha$ and derive $\mathcal{O}_0 \models \alpha \Leftrightarrow \mathcal{O}_n \models \alpha$ from it. In the loop iteration, either an axiom is moved from $\mathcal{O}$ to $\mathcal{O}'$ or $\mathcal{O} = (\mathcal{O} \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$ for some $\mathsf{R} \in \{\mathsf{NR1}, \ldots, \mathsf{NR6}\}$. Since in the former case, $\mathcal{O}_i = \mathcal{O}_{i+1}$ and $\mathcal{O}_i \models \alpha \Leftrightarrow \mathcal{O}_{i+1} \models \alpha$ follows trivially, we only cover the latter case, where $\mathcal{O}_{i+1} = (\mathcal{O}_i \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$.

$\mathcal{O}_i \models \alpha \Rightarrow \mathcal{O}_{i+1} \models \alpha$**,** for every axiom α.

We assume that $\mathcal{O}_i \models \alpha$ and we need to prove that $\mathcal{O}_{i+1} \models \alpha$. From $\mathcal{O}_i \models \alpha$ we have that for every interpretation $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{O}_i \Rightarrow \mathcal{I} \models \alpha \tag{3.11}$$

In order to prove $\mathcal{O}_{i+1} \models \alpha$, we assume that $\mathcal{I} \models \mathcal{O}_{i+1}$ and we prove $\mathcal{I} \models \alpha$. From Lemma 4 and since $\mathcal{O}_{i+1} = (\mathcal{O}_i \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$ and $\mathcal{I} \models \mathcal{O}_{i+1}$, we have that $\mathcal{I} \models \mathcal{O}_i$. Therefore, by (3.11) we derive $\mathcal{I} \models \alpha$ which was required to show.

$\mathcal{O}_{i+1} \models \alpha \Rightarrow \mathcal{O}_i \models \alpha$**,** for every axiom α that does not contain any fresh concept w.r.t. $\mathcal{O}_i$.

We assume that $\mathcal{O}_{i+1} \models \alpha$, that is for every interpretation $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{O}_{i+1} \Rightarrow \mathcal{I} \models \alpha \tag{3.12}$$

We want to prove that $\mathcal{O}_i \models \alpha$. Thus, we assume that $\mathcal{I} \models \mathcal{O}_i$ and we need to prove that $\mathcal{I} \models \alpha$. From Lemma 5 and since $\mathcal{I} \models \mathcal{O}_i$ and $\mathcal{O}_{i+1} = (\mathcal{O}_i \setminus \alpha') \cup \mathsf{R}(\alpha', \mathcal{O})$ we have that there exists an interpretation

$\mathcal{J}$ such that $\mathcal{I}|_{sig(\mathcal{O}_i)} = \mathcal{J}|_{sig(\mathcal{O}_i)}$ and $\mathcal{J} \models \mathcal{O}_{i+1}$. From (3.12), we derive $\mathcal{J} \models \alpha$. Since α does not contain any fresh concepts w.r.t. $\mathcal{O}_i$ and $\mathcal{I}|_{sig(\mathcal{O}_i)} = \mathcal{J}|_{sig(\mathcal{O}_i)}$, we have that $\mathcal{I} \models \alpha$, which was required to show.

□

Example 6. *The goal of this example is to show the necessity of the condition of Theorem 6 that α should not contain any fresh concepts w.r.t. $\mathcal{O}$. Let $\mathcal{O}_{ex}$ be the input and $\mathcal{O}'_{ex}$ the output of Algorithm 1 as shown in Example 1 (page 8) and Example 3 (page 12) respectively. For $\alpha = \exists \mathsf{hasSymptom}.\mathsf{Fever} \sqsubseteq \mathsf{A}$, we have that $\mathcal{O}' \models \alpha$, since $\alpha \in \mathcal{O}'$. We define the interpretation $\mathcal{I}$ by:*

$$\Delta^{\mathcal{I}} = \{x, y\}$$

$$\mathsf{MinorDosageAllowed}^{\mathcal{I}} = \{x\}$$

$$\mathsf{MinorDosagePrescribed}^{\mathcal{I}} = \{x\}$$

$$\mathsf{Fever}^{\mathcal{I}} = \{y\}$$

$$\mathsf{hasSymptom}^{\mathcal{I}} = \{(x, y)\}$$

$X^{\mathcal{I}} = \emptyset$ for $X \notin \{\mathsf{MinorDosageAllowed}, \mathsf{MinorDosageAllowed}, \mathsf{Fever}, \mathsf{hasSymptom}\}$

We have $\mathcal{I} \models \mathcal{O}$ because:

α_1 $\emptyset \subseteq \emptyset$

α_2 $\emptyset \subseteq \{x\}$

α_3 $\{x\} \cap \{x\} \subseteq \{x\}$

α_4 $\emptyset \subseteq \emptyset$

α_5 $\emptyset \subseteq \{x\}$

However, $\mathcal{I} \not\models \alpha$ because $(\exists \mathsf{hasSymptom}.\mathsf{Fever})^{\mathcal{I}} = \{x\} \not\subseteq \emptyset = \mathsf{A}^{\mathcal{I}}$. Therefore, $\mathcal{O} \not\models \alpha$, which is a consequence of the fact that α contains the fresh concept A.

Chapter 4

Saturation Stage and Complexity of Reasoning

In the current chapter, we present the second phase of the reasoning process, which is the saturation stage. Chapter 4 is divided to three sections. In Section 4.1 we provide preliminary definitions and notation which will be widely used from now on. In Section 4.2 we present a polynomial consequence-based reasoning algorithm and we prove in which cases it is sound and in which cases complete. Section 4.3 deals with the subsumption problem for some specific fragments of $\mathcal{EL}^{\bot}(D)$; for these cases subsumption problem proves to be EXPTIME-hard.

4.1 Numerical Datatypes with Operators for the $\mathcal{EL}$ Language

The present section introduces the notion of NDOs. An NDO can be perceived as a datatype with commonly used binary operators ($<$, $\leq$, $>$, $\geq$, $=$) along with restrictions on the use of the operators. Informally, an NDO determines which operators can be used at which side of an ontology's axioms.

Section 4.1.1 describes some connections that can be detected between D-datatype restrictions. Section 4.1.2 defines formally what a Numerical Datatype with Operators (NDO) is and when it is considered to be safe and convex.

4.1.1 Datatype Restrictions

We begin by defining when one or two datatype restrictions are inconsistent and when one datatype restriction implies another. The definitions are followed by a couple of examples, which clarify them.

Definition 7 (**Inconsistency for datatype restriction(s)**). *Let D be a numerical datatype and (op, q), (op', r) two D-datatype restrictions. A*

datatype restriction (op, q) *is defined to be* inconsistent w.r.t. a datatype D *(written* $(op, q) \rightarrow_D \bot$*) iff there is no* $x \in D$ *such that* x *satisfies* (op, q)*. Otherwise, we write* $(op, q) \nrightarrow_D \bot$*. Given that* $(op, q) \nrightarrow_D \bot$ *and* $(op', r) \nrightarrow_D \bot$*, we write* $(op, q) \wedge (op', r) \rightarrow_D \bot$ *and we say that* (op, q) *and* (op', r) *are* inconsistent w.r.t. a datatype D *iff there is no number* $x \in D$ *such that* x *satisfies* (op, q) *and* (op', r) *at the same time. Again, in the opposite case we write* $(op, q) \wedge (op', r) \nrightarrow_D \bot$*.*

Definition 8 (**Implication between datatype restrictions**)**.** *Let* D *be a numerical datatype and* (op, q)*,* (op', r) *two* D*-datatype restrictions. We say that* (op, q) implies (op', r) w.r.t. a datatype D *and we write* $(op, q) \rightarrow_D (op', r)$ *iff every number* $x \in D$*, which satisfies* (op, q)*, satisfies* (op', r) *as well; we also require* $(op, q) \nrightarrow_D \bot$*. Otherwise, we write* $(op, q) \nrightarrow_D (op', r)$*.*

Example 7. *Let* $D = \mathbb{R}^+$*, where* $\mathbb{R}^+$ *is the set of positive real numbers and* $(<, 5)$*,* $(\leq, 7)$*,* $(<, 1)$ *and* $(>, 8)$ *be datatype restrictions.* $x = 4$ *satisfies* $(<, 5)$ *because* $4 < 5$*, but* $x = 8$ *does not. Furthermore,* $(<, 5)$ *implies* $(\leq, 7)$ *w.r.t.* $\mathbb{R}^+$ *because for every* $x \in \mathbb{R}^+$*, if* $x < 5$ *then* $x \leq 7$ *as well. Additionally,* $(\leq, 7) \nrightarrow_{\mathbb{R}^+} (<, 5)$*, because e.g.,* $x = 6$ *satisfies* $(\leq, 7)$ *but not* $(<, 5)$*. Also,* $(<, 1) \rightarrow_{\mathbb{R}^+} \bot$ *and* $(<, 5) \wedge (>, 8) \rightarrow_{\mathbb{R}^+} \bot$*.*

Example 8. *Let* $D = \{2, 3, 4\}$ *and* $(>, 3)$*,* $(<, 3)$*,* $(\leq, 4)$*,* $(\leq, 2)$*,* $(<, 2)$ *and* $(=, 2)$ *be* D*-datatype restrictions. Then* $(>, 3)$ *implies* $(\leq, 4)$ *and* $(\leq, 2)$ *implies* $(=, 2)$ *w.r.t.* D*. Additionally,* $(<, 2) \rightarrow_D \bot$ *and* $(>, 2) \wedge (<, 3) \rightarrow_D \bot$*.*

Definition 9. *Let* D *be a numerical datatype. We write* $p = max(D)$ *if* $p \in D$ *and* $\nexists\ q \in D$ *such that* $q > p$*. We, further, write* $p = min(D)$ *if* $p \in D$ *and* $\nexists\ q \in D$ *such that* $q < p$*.*

Table 4.1 provides a complete listing of the cases where a datatype restriction is inconsistent w.r.t. a datatype D; the listing is complete: from the five different datatype restrictions only two can be inconsistent and are presented with the corresponding conditions.

Table 4.2 presents exhaustively the cases, where two datatype restrictions are inconsistent w.r.t. a datatype D modulo commutativity of conjunction between the datatype restrictions; there are no more cases to be included, because we present conditions for all possible pairs of datatype restrictions apart from the ones that do not imply inconsistency at any condition. Suppose that the first and second datatype restrictions define sets S_1, $S_2 \subseteq \mathbb{R}$; the condition given at Table 4.2 corresponds to disjointness between S_1 and S_2.

Finally, Table 4.3 gives a comprehensive list of datatype restrictions that imply other datatype restrictions and the necessary conditions for this to happen. Again, all cases are considered (25 in total) and the corresponding conditions are given. Again, if we assume that the first and second datatype

(op, q)	$(op, q) \rightarrow_D \bot$
$(<, q)$	$q = min(D)$
$(>, q)$	$q = max(D)$

Table 4.1: $(op, q) \rightarrow_D \bot$ cases

(op, q)	(op', r)	Condition for $(op, q) \wedge (op', r) \rightarrow_D \bot$
$(\leq, q)$	$(\geq, r)$	$\nexists\ p \in D$ s.t. $r \leq p$ and $p \leq q$
$(<, q)$	$(\geq, r)$	$\nexists\ p \in D$ s.t. $r \leq p$ and $p < q$
$(\leq, q)$	$(>, r)$	$\nexists\ p \in D$ s.t. $r < p$ and $p \leq q$
$(<, q)$	$(>, r)$	$\nexists\ p \in D$ s.t. $r < p$ and $p < q$
$(=, q)$	$(>, r)$	$q \leq r$
$(=, q)$	$(<, r)$	$q \geq r$
$(=, q)$	$(\geq, r)$	$q < r$
$(=, q)$	$(\leq, r)$	$q > r$
$(=, q)$	$(=, r)$	$q \neq r$

Table 4.2: $(op, q) \wedge (op', r) \rightarrow_D \bot$ cases

restrictions define sets S_1, $S_2 \subseteq \mathbb{R}$, the condition given at Table 4.3 is the condition such that $S_1 \subseteq S_2$.

Since there is case that one or two datatype restriction can be inconsistent w.r.t. a datatype D ($(op, q) \rightarrow_D \bot$ or $(op_1, q_1) \wedge (op_2, q_2) \rightarrow_D \bot$), one may wonder whether there is a similar case for three datatype restrictions. The answer is that there is not such a case, as proved by the following proposition. We provide a definition first which is used in the proof of the proposition.

Definition 10. *We say that two datatype restrictions have* the same direction *if their relational operators are one of the following: both $>$, both $\geq$, both $<$, both $\leq$, $>$ and $\geq$, $<$ and $\leq$. We say that two datatype restrictions have* opposite directions *in the remaining cases, excluding the cases where at least one operator is equality. The definition can be generalized for more than two datatype restrictions.*

Proposition 3. *If three datatype restrictions are inconsistent then at least a pair or one of them is inconsistent.*

Proof. We assume the opposite and prove that it is false. Let D be a datatype and r_1, r_2 and r_3 be three D-datatype restrictions, such that $r_i \nrightarrow_D \bot$ and $r_i \wedge r_j \nrightarrow_D \bot$ for $i, j = 1, 2, 3$ and $i \neq j$, and there is no $x \in D$ which satisfies r_1, r_2 and r_3 at the same time. We distinguish three cases:

- There is at least one equality, e.g., $r_1 = (=, a)$. Since $r_1 \wedge r_2 \nrightarrow_D \bot$

(op, q)	(op', r)	Condition for $(op, q) \rightarrow_D (op', r)$
$(\leq, q)$	$(\leq, r)$	$\nexists\, p \in D$ s.t. $p \leq q$ and $p > r$
$(\leq, q)$	$(<, r)$	$\nexists\, p \in D$ s.t. $p \leq q$ and $p \geq r$
$(<, q)$	$(\leq, r)$	$\nexists\, p \in D$ s.t. $p < q$ and $p > r$
$(<, q)$	$(<, r)$	$\nexists\, p \in D$ s.t. $p < q$ and $p \geq r$
$(\geq, q)$	$(\geq, r)$	$\nexists\, p \in D$ s.t. $p \geq q$ and $p < r$
$(\geq, q)$	$(>, r)$	$\nexists\, p \in D$ s.t. $p \geq q$ and $p \leq r$
$(>, q)$	$(\geq, r)$	$\nexists\, p \in D$ s.t. $p > q$ and $p < r$
$(>, q)$	$(>, r)$	$\nexists\, p \in D$ s.t. $p > q$ and $p \leq r$
$(=, q)$	$(\leq, r)$	$q \leq r$
$(=, q)$	$(<, r)$	$q < r$
$(=, q)$	$(\geq, r)$	$q \geq r$
$(=, q)$	$(>, r)$	$q > r$
$(=, q)$	$(=, r)$	$q = r$
$(>, q)$	$(=, r)$	$\nexists\, p \in D$ s.t. $p \neq r$ and $p > q$
$(<, q)$	$(=, r)$	$\nexists\, p \in D$ s.t. $p \neq r$ and $p < q$
$(>, q)$	$(\leq, r)$	$\nexists\, p \in D$ s.t. $p > q$ and $p > r$
$(>, q)$	$(<, r)$	$\nexists\, p \in D$ s.t. $p > q$ and $p \geq r$
$(\geq, q)$	$(\leq, r)$	$\nexists\, p \in D$ s.t. $p > q$ and $p > r$
$(\geq, q)$	$(<, r)$	$\nexists\, p \in D$ s.t. $p \geq q$ and $p \geq r$
$(\geq, q)$	$(=, r)$	$\nexists\, p \in D$ s.t. $p \geq q$ and $p \neq r$
$(<, q)$	$(\geq, r)$	$\nexists\, p \in D$ s.t. $p < q$ and $p < r$
$(<, q)$	$(>, r)$	$\nexists\, p \in D$ s.t. $p < q$ and $p \leq r$
$(\leq, q)$	$(\geq, r)$	$\nexists\, p \in D$ s.t. $p \leq q$ and $p < r$
$(\leq, q)$	$(>, r)$	$\nexists\, p \in D$ s.t. $p \leq q$ and $p \leq r$
$(\leq, q)$	$(=, r)$	$\nexists\, p \in D$ s.t. $p \leq q$ and $p \neq r$

Table 4.3: $(op, q) \rightarrow_D (op', r)$ cases

and $r_2 \wedge r_3 \nrightarrow_D \bot$, a satisfies both r_2 and r_3. Hence, there is an $x \in D$ which satisfies r_1, r_2 and r_3 at the same time.

- r_1, r_2 and r_3 have the same direction. W.l.o.g. we assume that $r_1 = (>, a)$, $r_2 = (>, b)$ and $r_3 = (>, c)$. Since $r_i \nrightarrow_D \bot$ for $i = 1, 2, 3$, there is an $x \in D$, such that $x > \max(a, b, c)$, which satisfies all the three of them.

- r_1, r_2 have the same direction and r_3 has direction opposite to them. W.l.o.g. we assume that $r_1 = (>, a)$, $r_2 = (>, b)$ and $r_3 = (<, c)$. Since $r_i \wedge r_j \nrightarrow_D \bot$ for $i, j = 1, 2, 3$ and $i \neq j$, there is an $x \in D$, such that $\max(a, b) < x < c$, which satisfies all the three of them.

  Thus, our initial assumption (that there is no $x \in D$ which satisfies r_1, r_2 and r_3 at the same time where $r_i \nrightarrow_D \bot$ and $r_i \wedge r_j \nrightarrow_D \bot$ for

$i, j = 1, 2, 3$ and $i \neq j$) was false and if, three datatype restrictions are inconsistent then at least one or a pair of them is inconsistent.

□

4.1.2 Safety and Convexity for NDOs

At this point, we describe a Numerical Datatype with Operators with a formal definition and an explanatory example.

Definition 11 (**Numerical Datatype with Operators(NDO)**). *We call the triple (D, O_-, O_+), where $D \subseteq \mathbb{R}$ and $O_-, O_+ \subseteq \{<, \leq, >, \geq, =\}$ a* numerical datatype with operators (NDO). *The $\mathcal{EL}^\perp(D, O_-, O_+)$ language is the $\mathcal{EL}^\perp(D)$ language, where for every positive occurrence of concepts of the form $\exists F.(op, q)$ (in axioms) there should be $op \in O_+$ and $q \in D$ and for every negative occurrence of concepts of the form $\exists F.(op, q)$ (in axioms) there should be $op \in O_-$ and $q \in D$. We say that an axiom is in $\mathcal{EL}^\perp(D, O_-, O_+)$ if it obeys the above restrictions.*

Intuitively, an NDO restricts the expressivity of $\mathcal{EL}^\perp(D)$ by defining precisely which operators can be used at the left-hand side of axioms (O_-) and which at the right-hand side of axioms (O_+).

Example 9. *$\mathcal{EL}(\mathbb{Z}, \{<, >\}, \{=\})$ permits the axioms $A \sqsubseteq \exists F.(=, 4)$, $\exists F.(<, 8) \sqsubseteq B$ or $\exists F.(>, 13) \sqsubseteq C$ but not the axioms $A \sqsubseteq \exists F.(>, 9)$, $\exists F.(<, 10.5) \sqsubseteq B$.*

Now, we define what we call from now on a constraint and when it is considered to be satisfiable. We also consider a separate case of functional constraints. The constraint proves to be a particularly useful structure at the definition of a safe NDO and during the completeness proof.

Definition 12 (**Satisfiability of a Constraint**). *Let (D, O_-, O_+) be an NDO. Let, also, $S^+ = \{(op_1, q_1), \ldots, (op_n, q_n)\}$ such that $op_1, \ldots, op_n \in O_+$, $q_1, \ldots, q_n \in D$ and $S^- = \{(op'_1, r_1), \ldots, (op'_m, r_m)\}$ such that $op'_1, \ldots, op'_m \in O_-$, $r_1, \ldots, r_m \in D$; we further require $(op_i, q_i) \nrightarrow_D (op'_j, r_j)$ and $(op_i, q_i) \nrightarrow_D \perp$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. We call the pair (S^+, S^-) a* constraint w.r.t. (D, O_-, O_+) *and we say that (S^+, S^-) is satisfiable w.r.t. D iff there exists a set $V \subseteq D$ such that every $r^+ \in S^+$ is satisfied by at least one $v \in V$ but no $r^- \in S^-$ is satisfied by any $v \in V$.*

Definition 13 (**Satisfiability of a Functional Constraint**). *Let (D, O_-, O_+) be an NDO. Let, also, $S^+ = \{(op_1, q_1), \ldots, (op_n, q_n)\}$ such that $op_1, \ldots, op_n \in O_+$, $q_1, \ldots, q_n \in D$ and $S^- = \{(op'_1, r_1), \ldots, (op'_m, r_m)\}$ such that $op'_1, \ldots, op'_m \in O_-$, $r_1, \ldots, r_m \in D$; we further require $(op_i, q_i) \nrightarrow_D (op'_j, r_j)$, $(op_i, q_i) \nrightarrow_D \perp$ and $(op_i, q_i) \wedge (op_l, q_l) \nrightarrow_D \perp$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$, $l = 1, \ldots, n$ and $l \neq i$. We call the pair (S^+, S^-) a* functional constraint

w.r.t. (D, O_-, O_+) *and we say that* (S^+, S^-) *is satisfiable w.r.t.* D *iff there exists a set* $V \subseteq D$ *such that* $|V| \leq 1$ *and every* $r^+ \in S^+$ *is satisfied by at least one* $v \in V$ *but no* $r^- \in S^-$ *is satisfied by any* $v \in V$.

At this point, we are going to introduce the notion of safety for NDOs. As we show later in full detail, if an NDO (D, O_-, O_+) is safe, then there is a polynomial, sound and complete classification algorithm for the $\mathcal{EL}^\bot(D, O_-, O_+)$ language. We further consider functional safety which is an even stricter property; its usefulness is clarified in the completeness proof.

Definition 14 (**Safe NDO**). *Let* (D, O_-, O_+) *be an NDO.* (D, O_-, O_+) *is safe iff every constraint w.r.t.* (D, O_-, O_+) *is satisfiable.*

Definition 15 (**Functionally Safe NDO**). *Let* (D, O_-, O_+) *be an NDO.* (D, O_-, O_+) *is safe iff every functional constraint w.r.t.* (D, O_-, O_+) *is satisfiable.*

Definition 16. *Let* D *be a numerical datatype. We say that* a conjunction of datatype restrictions implies a disjunction of datatype restriction w.r.t. a datatype D *and we write* $\bigwedge_{i=1}^{n} r_i \rightarrow_D \bigvee_{j=1}^{m} r'_j$ *iff for every* $x \in D$, *if* x *satisfies every* r_i *with* $i = 1, \ldots, n$, *then it satisfies at least one* r'_j *with* $j = 1, \ldots, m$.

Another property of NDOs which proves to be important is convexity [1]. We formally define it here.

Definition 17 (**Convex NDO**). *Let* (D, O_-, O_+) *be an NDO,* $op_i \in O_+$, $op'_j \in O_-$ *and* $q_i, r_j \in D$, *for* $i = 1, \ldots, n$ *and* $j = 1, \ldots, m$. (D, O_-, O_+) *is convex iff for* $\bigwedge_{i=1}^{n} (op_i, q_i) \rightarrow_D \bigvee_{j=1}^{m} (op'_j, r_j)$ *there exists* j, *such that* $1 \leq j \leq m$ *and* $\bigwedge_{i=1}^{n} (op_i, q_i) \rightarrow_D (op'_j, r_j)$.

We also formulate the condition that an NDO should satisfy in order to be non-convex. The non-convexity of an NDO is a significant property because in Section 4.3 we prove that the non-convexity of an NDO leads to intractability.

Definition 18 (**Violation Convexity Condition**). *Let* (D, O_-, O_+) *be an NDO,* $op_i \in O_+$, $op'_j \in O_-$ *and* $q_i, r_j \in D$, *for* $i = 1, \ldots, n$ *and* $j = 1, \ldots, m$. *The tuple* $((op_1, q_1), \ldots, (op_n, q_n), (op'_1, r_1), \ldots, (op'_1, r_m))$ *is called a violation convexity condition when the following conditions hold:*

$$\bigwedge_{i=1}^{n} (op_i, q_i) \rightarrow_D \bigvee_{j=1}^{m} (op'_j, r_j)$$

$$\bigwedge_{i=1}^{n} (op_i, q_i) \nrightarrow_D (op'_1, r_1)$$

$$\bigwedge_{i=1}^{n}(op_i, q_i) \nrightarrow_D (op'_2, r_2)$$

$$\vdots$$

$$\bigwedge_{i=1}^{n}(op_i, q_i) \nrightarrow_D (op'_m, r_m)$$

Proposition 4. *Let (D, O_-, O_+) be an NDO. There is a violation convexity condition for (D, O_-, O_+) iff (D, O_-, O_+) is not convex.*

Proof. If we negate the condition from the definition of convexity we directly derive the violation convexity condition. □

Now we prove that safety implies convexity. Thus, safety is a property stronger and preciser than convexity, since a safe NDO is always convex.

Proposition 5. *Let (D, O_-, O_+) be an NDO. If (D, O_-, O_+) is safe, then it is convex as well.*

Proof. We proceed by proving the contrapositive, that is if (D, O_-, O_+) is not convex then it is not safe. In order to prove that (D, O_-, O_+) is not safe, it is sufficient to prove that there is a constraint (S^+, S^-) w.r.t. (D, O_-, O_+) such that (S^+, S^-) is not satisfiable.

According to 4, since (D, O_-, O_+) is not convex, there exists a violation convexity condition such that, for $op_i \in O_+$, $op'_j \in O_-$ and $q_i, r_j \in D$, with $i = 1, \ldots, n$ and $j = 1, \ldots, m$, the following conditions hold:

$$\bigwedge_{i=1}^{n}(op_i, q_i) \rightarrow_D \bigvee_{j=1}^{m}(op'_j, r_j)$$

$$\bigwedge_{i=1}^{n}(op_i, q_i) \nrightarrow_D (op'_1, r_1)$$

$$\bigwedge_{i=1}^{n}(op_i, q_i) \nrightarrow_D (op'_2, r_2)$$

$$\vdots$$

$$\bigwedge_{i=1}^{n}(op_i, q_i) \nrightarrow_D (op'_m, r_m)$$

We form the constraint (S^+, S^-) with $S^+ = \{(op_1, q_1), \ldots, (op_n, q_n)\}$ and $S^- = \{(op'_1, r_1), \ldots, (op'_1, r_m)\}$. The above (S^+, S^-) is indeed a constraint because according to Definition 12 we need $(op_i, q_i) \nrightarrow_D (op'_j, r_j)$, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$, which is a consequence of the violation convexity condition. (S^+, S^-) is not satisfiable because (from Definition 12) there

can be found no $v \in D$ such that it satisfies $\bigwedge_{i=1}^{n}(op_i, q_i)$ but none of the (op'_j, r_j) for $j = 1, \ldots, m$ as a result from $\bigwedge_{i=1}^{n}(op_i, q_i) \rightarrow_D \bigvee_{j=1}^{m}(op'_j, r_j)$ which is included in the violation convexity condition. □

4.2 Rules and Saturation Algorithm

The current section depicts the saturation stage, which is the second part of the reasoning process. Section 4.2.1 provides some consequence-based rules which lie at the heart of the saturation algorithm which is described in Section 4.2.2. Section 4.2.3. specifies when the algorithm is sound, when it is complete and supplies the accompanying proofs. The notation and definitions provided in Section 4.1 are widely used here.

4.2.1 Consequence-based Rules

Firstly, we present in Table 4.4 the rules IR1-IR2 and CR1-CR5 [1] which do not involve any datatypes and are essential for the saturation algorithm, which will be presented in the subsequent section. For the rules IR1-IR2, and CR1-CR5 we have that $A, B, C, D \in N_C^{\top}$, $C' \in N_C^{\top} \cup \{\bot\}$ and $r \in N_R$. The premise(s) of each rule is (are) over the line, the conclusion is under the line and on the right appear the side conditions, which are necessary to be satisfied for the rule to be applied.

Table 4.5 brings in the rules ID1-CD3, which deal with the datatype part of the language. For ID1-CD3 it holds that $A, B \in N_C^{\top}$ and $F \in N_F$. We assume that there exists a procedure based on Table 4.1, Table 4.2 and Table 4.3 that checks in polynomial time whether $(op, q) \rightarrow_D \bot$, $(op, q) \wedge (op', r) \rightarrow_D \bot$ and $(op, q) \rightarrow_D (op', r)$ correspondingly.

4.2.2 The Saturation Algorithm

After presenting the rules, we are ready to continue with the description of the saturation algorithm. Before that we define some notation which is used in the algorithm.

Definition 19. *Let $\mathcal{O}$ be an ontology, $\mathsf{R} \in \{\mathsf{CR1}, \ldots, \mathsf{CD3}\}$, $\alpha \in Ax$ and $S \subseteq Ax$. The function $\mathsf{R}(\mathcal{O}, \alpha, S)$ returns a set of axioms which is the union of the conclusions we obtain if we apply rule R, with α as one premise and with a member of S as the other premise (only for rules CR2, CR4, CR5 and CD3); the first argument, $\mathcal{O}$, is used in the side condition(s), if there exist any, which should be satisfied for the rules to be applicable (Table 4.4, Table 4.5).*

Rule	Premises-Conclusion	Side condition
IR1	$\frac{}{A \sqsubseteq A}$	–
IR2	$\frac{}{A \sqsubseteq \top}$	–
CR1	$\frac{A \sqsubseteq B}{A \sqsubseteq C'}$	$B \sqsubseteq C' \in \mathcal{O}$
CR2	$\frac{A \sqsubseteq B \quad A \sqsubseteq C}{A \sqsubseteq D}$	$B \sqcap C \sqsubseteq D \in \mathcal{O}$
CR3	$\frac{A \sqsubseteq B}{A \sqsubseteq \exists r.C}$	$B \sqsubseteq \exists r.C \in \mathcal{O}$
CR4	$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq C}{A \sqsubseteq D}$	$\exists r.C \sqsubseteq D \in \mathcal{O}$
CR5	$\frac{A \sqsubseteq \exists r.B \quad B \sqsubseteq \bot}{A \sqsubseteq \bot}$	-

Table 4.4: Reasoning rules in $\mathcal{EL}^{\bot}(D)$ (no datatypes)

Example 10. *Let* $\mathcal{O} = \{B \sqcap C \sqsubseteq D, \exists r.C \sqsubseteq D\}$ *and* $S = \{A \sqsubseteq C, B \sqsubseteq C\}$. *In that case, we have* $\mathsf{CR2}(A \sqsubseteq B, S) = \mathsf{CR4}(A \sqsubseteq \exists r.B, S) = \{A \sqsubseteq D\}$.

Algorithm 2 takes as input an ontology $\mathcal{O}$ of axioms in normal form and produces as output the ontology $\mathcal{O}'$; a taxonomy of $\mathcal{O}$ can be extracted from $\mathcal{O}'$ by retaining only axioms of the form NF1 ($A \sqsubseteq B$) and NF7 ($A \sqsubseteq \bot$). Algorithm 2 follows the pattern of the Given Clause Algorithm [15] and splits the axioms into two sets: the set of unprocessed ($\mathcal{O}'_1$) and processed ($\mathcal{O}'_2$) axioms. $\mathcal{O}'_1$ is initialized with the rules that do not require any premises (IR1, IR2, ID1). After that, every axiom α in $\mathcal{O}'_1$ is examined and all the rules are checked; at least one premise of the rule should be α and the other one (if there exists one) is taken from $\mathcal{O}'_2$. The new conclusions are checked if they occur in $\mathcal{O}'_1 \cup \mathcal{O}'_2$ and if not they are added to $\mathcal{O}'_1$; α is now moved from the set of unprocessed ($\mathcal{O}'_1$) to the set of processed ($\mathcal{O}'_2$) axioms. The algorithm

Rule	Premises-Conclusion	Side-condition(s)
ID1	$\dfrac{}{A \sqsubseteq \bot}$	$A \sqsubseteq \exists F.(op, q) \in \mathcal{O}$ $\wedge\ (op, q) \rightarrow_D \bot$
CD1	$\dfrac{A \sqsubseteq \exists F.(op, q)}{A \sqsubseteq B}$	$\exists F.(op', r) \sqsubseteq B \in \mathcal{O}$ $\wedge\ (op, q) \rightarrow_D (op', r)$
CD2	$\dfrac{A \sqsubseteq B}{A \sqsubseteq \exists F.(op, q)}$	$B \sqsubseteq \exists F.(op, q) \in \mathcal{O}$
CD3	$\dfrac{A \sqsubseteq \exists F.(op, q) \quad A \sqsubseteq \exists F.(op', r)}{A \sqsubseteq \bot}$	$(op, q) \wedge (op', r) \rightarrow_D \bot$ $\wedge\ \mathsf{Funct}(F) \in \mathcal{O}$

Table 4.5: Reasoning rules in $\mathcal{EL}^{\bot}(D)$ (for datatypes)

terminates when the set of unprocessed axioms ($\mathcal{O}'_1$) becomes empty.

Example 11. *In order to clarify Algorithm 2 we apply it to $\mathcal{O}'$, which was computed in Example 3 (page 12). We set $\mathcal{O} = \mathcal{O}'$ as the input of Algorithm 2 to avoid confusion. We do not show the detailed execution of the algorithm; we only show the axioms that the output ($\mathcal{O}' = \mathcal{O}'_1 \cup \mathcal{O}'_2$) contains and which rules produced them. We first present the axioms which were added to $\mathcal{O}'_1$ during initialization.*

α_6	$\mathsf{Child} \sqsubseteq \top$	α_7	$\mathsf{Feverish} \sqsubseteq \top$
α_8	$\mathsf{FeverishChild} \sqsubseteq \top$	α_9	$\mathsf{Fever} \sqsubseteq \top$
α_{10}	$\mathsf{A} \sqsubseteq \top$	α_{11}	$\mathsf{MinorDosagePrescribed} \sqsubseteq \top$
α_{12}	$\mathsf{MinorDosageAllowed} \sqsubseteq \top$	α_{13}	$\mathsf{Child} \sqsubseteq \mathsf{Child}$
α_{14}	$\mathsf{Feverish} \sqsubseteq \mathsf{Feverish}$	α_{15}	$\mathsf{FeverishChild} \sqsubseteq \mathsf{FeverishChild}$
α_{16}	$\mathsf{Fever} \sqsubseteq \mathsf{Fever}$	α_{17}	$\mathsf{A} \sqsubseteq \mathsf{A}$

Algorithm 2 : Saturation Algorithm

Input: = ontology $\mathcal{O}$ consisting of axioms in normal form
Output: = ontology $\mathcal{O}'$ consisting of axioms in NF1, NF3, NF5 and NF7

1: $\mathcal{O}'_1 := \{A \sqsubseteq A, A \sqsubseteq \top \mid A \in N_C \cap sig(\mathcal{O})\}$; \\IR1, IR2
2: $\mathcal{O}'_1 := \mathcal{O}'_1 \cup \{A \sqsubseteq \bot \mid A \sqsubseteq \exists F.(op, q) \in \mathcal{O} \wedge (op, q) \rightarrow_D \bot\}$; \\ID1
3: $\mathcal{O}'_2 := \emptyset$;
4: **repeat**
5: choose an axiom α such that $\alpha \in \mathcal{O}'_1$;
6: $U := \bigcup_{\mathsf{R} \in \mathsf{CR1-CR5,CD1-CD3}} \mathsf{R}(\mathcal{O}, \alpha, \mathcal{O}'_2)$;
7: **for** any $\alpha' \in U$ such that $\alpha' \notin \mathcal{O}'_1 \cup \mathcal{O}'_2$ **do**
8: $\mathcal{O}'_1 := \mathcal{O}'_1 \cup \{\alpha'\}$;
9: **end for**
10: $\mathcal{O}'_1 := \mathcal{O}'_1 \setminus \{\alpha\}$;
11: $\mathcal{O}'_2 := \mathcal{O}'_2 \cup \{\alpha\}$;
12: **until** $\mathcal{O}'_1 := \emptyset$;
13: $\mathcal{O}' := \mathcal{O}'_2$;

α_{18} MinorDosagePrescribed $\sqsubseteq$ MinorDosagePrescribed

α_{19} MinorDosageAllowed $\sqsubseteq$ MinorDosageAllowed

Now we show the axioms which were added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$ during the loop iterations, as a result of the rules CR1-CR5*,* CD1-CD3*:*

α_{20} Child $\sqsubseteq \exists$hasAge.$(<, 12)$
after application of CD2 *to* α_{13} *and* $\alpha_{1a} \in \mathcal{O}$

α_{21} Child $\sqsubseteq \exists$hasAge.$(>, 5)$
after application of CD2 *to* α_{13} *and* $\alpha_{1b} \in \mathcal{O}$

α_{22} FeverishChild $\sqsubseteq$ Child
after application of CR1 *to* α_{15} *and* $\alpha_{4a} \in \mathcal{O}$

α_{23} FeverishChild $\sqsubseteq$ Feverish
after application of CR1 *to* α_{15} *and* $\alpha_{4b} \in \mathcal{O}$

α_{24} Feverish $\sqsubseteq \exists$hasSymptom.Fever
after application of CR3 *to* α_{14} *and* $\alpha_5 \in \mathcal{O}$

α_{25} Child $\sqsubseteq$ MinorDosageAllowed
after application of CD1 *to* α_{20} *and* $\alpha_2 \in \mathcal{O}$, $(<, 12) \rightarrow_D (<, 18)$

α_{26} $\mathsf{FeverishChild} \sqsubseteq \exists \mathsf{hasAge}.(<, 12)$
after application of CD2 *to* α_{22} *and* $\alpha_{1a} \in \mathcal{O}$

α_{27} $\mathsf{FeverishChild} \sqsubseteq \exists \mathsf{hasAge}.(>, 5)$
after application of CD2 *to* α_{22} *and* $\alpha_{1b} \in \mathcal{O}$

α_{28} $\mathsf{FeverishChild} \sqsubseteq \exists \mathsf{hasSymptom}.\mathsf{Fever}$
after application of CR3 *to* α_{23} *and* $\alpha_5 \in \mathcal{O}$

α_{29} $\mathsf{Feverish} \sqsubseteq \mathsf{A}$
after application of CR4 *to* α_{24}, α_{16} *and* $\alpha_{3a} \in \mathcal{O}$

α_{30} $\mathsf{FeverishChild} \sqsubseteq \mathsf{MinorDosageAllowed}$
after application of CD1 *to* α_{26} *and* $\alpha_2 \in \mathcal{O}$, $(<, 12) \rightarrow_D (<, 18)$

α_{31} $\mathsf{FeverishChild} \sqsubseteq \mathsf{A}$
after application of CR4 *to* α_{28}, α_{16} *and* $\alpha_{3a} \in \mathcal{O}$

α_{32} $\mathsf{FeverishChild} \sqsubseteq \mathsf{MinorDosagePrescribed}$
after application of CR2 *to* α_{30}, α_{31} *and* $\alpha_{3b} \in \mathcal{O}$

For $D' = \{r \in \mathbb{Z} \mid 12 \leq r \leq 120\}$ *we would have* $\mathsf{Child} \sqsubseteq \bot$ *as a result of the application of* ID1 *and* $\alpha_{1a} \in \mathcal{O}$, $(<, 12) \rightarrow_{D'} \bot$. *Additionally, if* $\mathsf{Child} \sqsubseteq \exists \mathsf{hasAge}.\ (>, 12) \in \mathcal{O}$ *and* $\mathsf{Funct}(F) \in \mathcal{O}$, $Child \sqsubseteq \bot$ *would have been added as a result of the application of* CD3.

We now prove that the algorithm terminates and after that we prove that the algorithm costs polynomial space and polynomial time.

Proposition 6 (**Termination of Saturation Algorithm**). *Algorithm 2 terminates.*

Proof. We prove termination by proving that the loop of Algorithm 2 iterates finitely many times. It is sufficient to prove that the conditional assignment $\mathcal{O}'_1 := \mathcal{O}'_1 \cup \{\alpha'\}$ is executed a finite number of times. Thus, $\mathcal{O}'_1$ contains finitely many axioms and since one axiom is unconditionally removed from $\mathcal{O}'_1$ per loop iteration, Algorithm 2 terminates.

Now we prove that $\mathcal{O}'_1 := \mathcal{O}'_1 \cup \{\alpha'\}$ is executed finitely many times. According to the algorithm, an axiom α' is added to $\mathcal{O}'_1$ if it is a conclusion of one of the rules CR1-CD3 and does not belong to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. The number of possible conclusions of CR1-CD3 is bounded because they are axioms in normal form and formed using symbols from the finite set $sig(\mathcal{O})$. Furthermore, once an axiom is added to $\mathcal{O}'_1$, either it remains there or it is moved to $\mathcal{O}'_2$. Since there is a check for duplicate axioms in $\mathcal{O}'_1 \cup \mathcal{O}'_2$, after finitely many additions of axioms to $\mathcal{O}'_1$, no more axiom is added to it.

□

Proposition 7 (Complexity of Saturation Algorithm). *Algorithm 2 runs in polynomial space and polynomial time.*

Proof. We set the finite sets (since $\mathcal{O}$ is finite) $N'_C = N_C^{\top} \cap sig(\mathcal{O})$, $N'_R = N_R \cap sig(\mathcal{O})$ and $N'_F = N_F \cap sig(\mathcal{O})$. Suppose that $|N'_C| = n$, $|N'_R| = m$ and $\mathcal{O}$ contains k axioms.

Claim 4. *Algorithm 2 runs in polynomial space.*

We calculate the maximum number of axioms that Algorithm 2 adds to $\mathcal{O}'_2$, which is its output. Axioms are added to $\mathcal{O}'_2$ only if they are produced by one of the IR1-IR2, CR1-CR5, ID1, CD1-CD3. Thus, only axioms of the form $A \sqsubseteq B$, $A \sqsubseteq \top$, $A \sqsubseteq \bot$, $A \sqsubseteq \exists r.B$ or $A \sqsubseteq \exists F.(op, q)$ with A, $B \in N'_C$, $r \in N'_R$ and $F \in N'_F$ are added. Since no fresh concepts, roles or features are introduced, N'_C, N'_R and N'_F remain the same. Furthermore, there is a check for the axioms that have already been added to $\mathcal{O}'_2$ not to be added again. We count how many different axioms there are at most for each form:

- $A \sqsubseteq B$, A, $B \in N_C$: n^2 axioms
- $A \sqsubseteq \top$, $A \in N_C$: n axioms
- $A \sqsubseteq \bot$, $A \in N_C$: n axioms
- $A \sqsubseteq \exists r.B$, A, $B \in N_C$, $r \in N_R$: $n^2 m$ axioms
- $A \sqsubseteq \exists F.(op,\ q)$, $A \in N_C$, $\exists F.(op, q)$ occurs in $\mathcal{O}$: nk axioms

In total, the maximum size of $\mathcal{O}'$ (that is the size of $\mathcal{O}'_2$ after the last iteration loop) is $2n+n^2(m+1)+kn = n(2+nm+n+k)$. As a consequence Algorithm 2 runs in $O(n^2 m + n \cdot k)$ space.

Claim 5. *Algorithm 2 runs in polynomial time.*

We set the maximum size of $\mathcal{O}'$, $l = n(2+nm+n+k)$. Let also C_{imp}, C_{inc_1} and C_{inc_2} be the cost (polynomial as we have assumed) for checking whether $(op, q) \rightarrow_D (op', r)$, $(op, q) \rightarrow_D \bot$ and $(op, q) \wedge (op', r) \rightarrow_D \bot$ respectively. During the initialization the algorithm runs in $O(2n + k \cdot C_{inc_1})$. Now, we calculate the worst-case time cost by multiplying:

- the upper bound of loop iterations

  The loop cannot be executed more than l times which is the upper space limit for $\mathcal{O}'_1$, since one axiom is unconditionally removed from $\mathcal{O}'_1$ per iteration.

- the worst-case cost per loop

  It is the sum of:

- time to calculate $\mathsf{CR1}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ ($\mathsf{CR3}(\mathcal{O}, \alpha, \mathcal{O}'_2)$) which is the size of $\mathcal{O}$, i.e., $O(k)$.
- time to calculate $\mathsf{CR2}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ ($\mathsf{CR4}(\mathcal{O}, \alpha, \mathcal{O}'_2)$) which is the size of $\mathcal{O}$ multiplied by the maximum size of $\mathcal{O}'_2$, i.e., $O(k \cdot l)$.
- time to calculate $\mathsf{CR5}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ which is the maximum size of $\mathcal{O}'_2$, i.e., $O(l)$.
- time to calculate $\mathsf{CD1}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ which is the size of $\mathcal{O}$ and the cost to check implication, i.e., $O(k \cdot C_{imp})$.
- time to calculate $\mathsf{CD2}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ which is the size of $\mathcal{O}$, i.e., $O(k)$.
- time to calculate $\mathsf{CD3}(\mathcal{O}, \alpha, \mathcal{O}'_2)$ which is the maximum size of $\mathcal{O}'_2$ times the cost to check inconsistency for two datatype restrictions, i.e., $O(l \cdot C_{inc_2})$.

In total, we have a polynomial upper time bound, which is $O(2n+k\cdot C_{inc_1})+ l \cdot O(k + kl + l + k \cdot C_{imp} + l \cdot C_{inc_2})$. If we further assume that C_{imp}, C_{inc_1} and C_{inc_2} run in constant time we finally find that Algorithm 2 runs in $O(k \cdot n^4 \cdot m^2 + n^2 \cdot k^3)$.

□

4.2.3 Soundness and Completeness

Now we continue with the soundness and completeness proof for Algorithm 2. The soundness means that the axioms contained in the output ontology are entailed by the input ontology, while completeness guarantees that if an axiom of the form $A \sqsubseteq B$ is entailed by the input ontology, then $A \sqsubseteq B$ or $A \sqsubseteq \bot$ is included in the output ontology. We prove that Algorithm 2 is sound without any further restriction. However, for the completeness we require that the axioms in the input ontology $\mathcal{O}$ be in $\mathcal{EL}^{\bot}(D, O_-, O_+)$, where (D, O_-, O_+) is a (functionally) safe NDO.

Theorem 20 (**Soundness for Saturation Algorithm**). *Let $\mathcal{O}$ be an ontology in $\mathcal{EL}^{\bot}(D)$ consisting of axioms in normal form and $\mathcal{O}'$ the ontology we obtain after applying Algorithm 2 to $\mathcal{O}$. Every model $\mathcal{I}$ of $\mathcal{O}$ is a model of $\mathcal{O}'$ as well.*

Proof. After the termination of the algorithm, we have that $\mathcal{O}' = \mathcal{O}'_1 \cup \mathcal{O}'_2$. Hence, we prove our lemma using induction on the size of $\mathcal{O}'_1 \cup \mathcal{O}'_2$. We consider as base case the ontology $\mathcal{O}'_1$ after the initialization (lines 1-3) of the algorithm (since $\mathcal{O}'_2$ is empty at that point). The induction case is the addition of axioms to $\mathcal{O}'_1$ during the loop (lines 4-12), since the axioms are first added to $\mathcal{O}'_1$ and then moved to $\mathcal{O}'_2$.

- Base case

Every model $\mathcal{I}$ of $\mathcal{O}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$, where $\mathcal{O}'_1 \cup \mathcal{O}'_2 = \{A \sqsubseteq A, A \sqsubseteq \top \mid A \in N_C \cap sig(\mathcal{O})\} \cup \{A \sqsubseteq \bot \mid A \sqsubseteq \exists F.(op, q) \in \mathcal{O} \wedge (op, q) \rightarrow_D \bot\}$.

If $\mathcal{I}$ is a model of $\mathcal{O}$, then $\mathcal{I}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$ as well, because $A^{\mathcal{I}} \subseteq A^{\mathcal{I}}$ and $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (for $A \in N_C \cap sig(\mathcal{O})$) hold trivially. For the axioms of the form $A \sqsubseteq \bot$ we have to show that $A^{\mathcal{I}} \subseteq \emptyset$. Since, $A \sqsubseteq \exists F.(op, q) \in \mathcal{O}$ and $\mathcal{I}$ is a model of $\mathcal{O}$, $A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$. However, from $(op, q) \rightarrow_D \bot$, we have that there exists no $x \in D$, such that it satisfies (op, q) and, as a result, $(\exists F.(op, q))^{\mathcal{I}} = \emptyset$.

- Induction step

Let $\mathcal{I}$ be a model of $\mathcal{O}$. If $\mathcal{I}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$, then it is also a model of $(\mathcal{O}'_1 \cup \mathcal{O}'_2) \cup \{\alpha'\}$, with $\alpha' \in \mathsf{R}(\mathcal{O}, \alpha, \mathcal{O}'_2)$, where $\mathsf{R} \in \mathsf{CR1} - \mathsf{CD3}$, $\alpha' \notin \mathcal{O}'_1 \cup \mathcal{O}'_2$ and $\alpha \in \mathcal{O}'_1$.

Since $\mathcal{I}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$, it is only left to show that $\mathcal{I} \models \alpha'$ for $\alpha' \in \mathsf{R}(\mathcal{O}, \alpha, \mathcal{O}'_2)$, where $\mathsf{R} \in \mathsf{CR1} - \mathsf{CD3}$, $\alpha' \notin \mathcal{O}'_1 \cup \mathcal{O}'_2$ and $\alpha \in \mathcal{O}'_1$. We distinguish cases:

CR1 In that case, $\alpha' = A \sqsubseteq C'$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be a $B \in N_C^{\top}$, such that $A \sqsubseteq B \in \mathcal{O}'_1$ and $B \sqsubseteq C' \in \mathcal{O}$. Since $\mathcal{I}$ is a model both of $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq B$ and $\mathcal{I} \models B \sqsubseteq C'$. Hence, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$, which gives us $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. Thus, $\mathcal{I} \models A \sqsubseteq C'$, which we needed to prove.

CR2 In that case, $\alpha' = A \sqsubseteq D$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be $B, C \in N_C^{\top}$, such that $A \sqsubseteq B \in \mathcal{O}'_1$, $A \sqsubseteq C \in \mathcal{O}'_2$ and $B \sqcap C \sqsubseteq D \in \mathcal{O}$. Since $\mathcal{I}$ is a model both of $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq B$, $\mathcal{I} \models A \sqsubseteq C$ and $\mathcal{I} \models B \sqcap C \sqsubseteq D$. Hence, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and $B^{\mathcal{I}} \cap C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, which gives us $A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Thus, $\mathcal{I} \models A \sqsubseteq D$, which is what we wanted to show.

CR3 In that case, $\alpha' = A \sqsubseteq \exists r.C$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be a $B \in N_C^{\top}$, such that $A \sqsubseteq B \in \mathcal{O}'_1$ and $B \sqsubseteq \exists r.C \in \mathcal{O}$. Since $\mathcal{I}$ is a model of both $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq B$ and $\mathcal{I} \models B \sqsubseteq \exists r.C$. Hence, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq (\exists r.C)^{\mathcal{I}}$, which gives us $A^{\mathcal{I}} \subseteq (\exists r.C)^{\mathcal{I}}$. Thus $\mathcal{I} \models A \sqsubseteq \exists r.C$, which was required to prove.

CR4 In that case, $\alpha' = A \sqsubseteq E$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be B, $C \in N_C^{\top}$ and $r \in N_R$, such that $A \sqsubseteq \exists r.B \in \mathcal{O}'_1$, $B \sqsubseteq C \in \mathcal{O}'_2$ (or $A \sqsubseteq \exists r.B \in \mathcal{O}'_2$, $B \sqsubseteq C \in \mathcal{O}'_1$) and $\exists r.C \sqsubseteq E \in \mathcal{O}$. Since $\mathcal{I}$ is a model both of $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq \exists r.B$, $\mathcal{I} \models B \sqsubseteq C$ and $\mathcal{I} \models \exists r.C \sqsubseteq E$. Hence, $A^{\mathcal{I}} \subseteq (\exists r.B)^{\mathcal{I}}$, $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ and $(\exists r.C)^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. Take an $x \in A^{\mathcal{I}}$. In order to show $A^{\mathcal{I}} \subseteq E^{\mathcal{I}}$, we prove $x \in E^{\mathcal{I}}$. For every $x \in A^{\mathcal{I}}$ there exists a y such that $(x, y) \in r^{\mathcal{I}}$ and $y \in B^{\mathcal{I}}$. Since $y \in B^{\mathcal{I}}$, then $y \in C^{\mathcal{I}}$. Furthermore, for every z such

that $(z, y) \in r^{\mathcal{I}}$ we have that $z \in E^{\mathcal{I}}$ and since $(x, y) \in r^{\mathcal{I}}$, we have that $x \in E^{\mathcal{I}}$. Thus, $\mathcal{I} \models A \sqsubseteq E$, which we needed to show.

CR5 In that case, $\alpha' = A \sqsubseteq \bot$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be an $r \in N_R$ and $B \in N_C^{\top}$, such that $A \sqsubseteq \exists r.B \in \mathcal{O}'_1$ and $B \sqsubseteq \bot \in \mathcal{O}'_2$ (or $A \sqsubseteq \exists r.B \in \mathcal{O}'_2$ and $B \sqsubseteq \bot \in \mathcal{O}'_1$). Since $\mathcal{I}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq \exists r.B$, $\mathcal{I} \models B \sqsubseteq \bot$. Hence, $A^{\mathcal{I}} \subseteq (\exists r.B)^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq (\bot)^{\mathcal{I}}$. From the semantics of bottom concept, $(\bot)^{\mathcal{I}} = \emptyset$ and, as a consequence, $B^{\mathcal{I}} = \emptyset$. From the interpretation of existential restriction, we have that $(\exists r.B)^{\mathcal{I}} = \emptyset$. Since $A^{\mathcal{I}} \subseteq (\exists r.B)^{\mathcal{I}}$ and $(\exists r.B)^{\mathcal{I}} = \emptyset$, $A^{\mathcal{I}} \subseteq \emptyset$. Thus, $A \sqsubseteq \bot$, which we needed to show.

CD1 In that case, $\alpha' = A \sqsubseteq B$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be an $F \in N_F$, such that $A \sqsubseteq \exists F.(op, q) \in \mathcal{O}'_1$ and $\exists F.(op', r) \sqsubseteq B \in \mathcal{O}$ with $(op, q) \rightarrow_D (op', r)$. Since $\mathcal{I}$ is a model both of $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq \exists F.(op, q)$ and $\mathcal{I} \models \exists F.(op', r) \sqsubseteq B$. Hence $A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$ and $(\exists F.(op', r))^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. However, since $(op, q) \rightarrow_D (op', r)$, every $x \in D$ that satisfies (op, q), satisfies (op', r) as well. Hence, $(\exists F.(op, q))^{\mathcal{I}} \subseteq (\exists F.(op', r))^{\mathcal{I}}$ and, thus, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$, which we wanted to show.

CD2 In that case, $\alpha' = A \sqsubseteq \exists F.(op, q)$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be a $B \in N_C^{\top}$, such that $A \sqsubseteq B \in \mathcal{O}'_1$ and $B \sqsubseteq \exists F.(op, q) \in \mathcal{O}$. Since $\mathcal{I}$ is a model both of $\mathcal{O}$ and $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq B$ and $\mathcal{I} \models B \sqsubseteq \exists F.(op, q)$. Hence, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$. We directly derive that $A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$.

CD3 In that case, $\alpha' = A \sqsubseteq \bot$ has been added to $\mathcal{O}'_1 \cup \mathcal{O}'_2$. Therefore, there should be an $F \in N_F$ such that $A \sqsubseteq \exists F.(op, q) \in \mathcal{O}'_1$, $A \sqsubseteq \exists F.(op', r) \in \mathcal{O}'_2$ with $(op, q) \wedge (op', r) \rightarrow_D \bot$ and $\mathsf{Funct}(F) \in \mathcal{O}$. Since $\mathcal{I}$ is a model of $\mathcal{O}'_1 \cup \mathcal{O}'_2$, we have that $\mathcal{I} \models A \sqsubseteq \exists F.(op, q)$ and $\mathcal{I} \models A \sqsubseteq \exists F.(op', r)$. Hence $A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$ and $A^{\mathcal{I}} \subseteq (\exists F.(op', r))^{\mathcal{I}}$. However, given that $(op, q) \wedge (op', r) \rightarrow_D \bot$ there is no $v \in D$, which satisfies at the same time (op, q) and (op', r). Therefore, since F is a functional feature, there is no x, such that $x \in (\exists F.(op, q))^{\mathcal{I}}$ and $x \in (\exists F.(op', r))^{\mathcal{I}}$ and, thus, $A^{\mathcal{I}} \subseteq \emptyset$.

□

The following lemma is later used in the completeness proof. It intuitively says that if the axioms of an input ontology $\mathcal{O}$ are in $\mathcal{EL}^{\bot}(D, O_-, O_+)$, then the axioms of the ontology obtained after applying Algorithms 1 and 2 to the input ontology are in $\mathcal{EL}^{\bot}(D, O_-, O_+)$ as well.

Lemma 6. *Let $\mathcal{O}$ be an ontology and $\mathcal{O}'$ the ontology obtained after applying Algorithm 1 to $\mathcal{O}$ and subsequently Algorithm 2 to the output of Algorithm 1.*

A concept of the form $\exists F.(op, q)$ positively (negatively) occurs in $\mathcal{O}'$ only if it positively (negatively) occurs in $\mathcal{O}$.

Proof. It suffices to prove that the normalization rules (NR1-NR5) and saturation rules (IR1-CR3) preserve the positive(negative) occurrences of existential datatype restrictions. For the rules NR1-NR5 (table 3.2 on page 12) a concept C occurs positively (negatively) in the head of a rule if it occurs positively (negatively) in one of the axioms of its body. For positive occurrences, this is the case for concepts E of NR1, D of NR2 and NR5, H of NR3 and NR4 and C of NR5. For negative occurrences, this is the case for concepts C and H of NR1, G of NR2 and NR3 and C of NR4. The only saturation rule which adds an axiom with a positive occurrence of $\exists F.(op, q)$ is CR3, where a concept of the form $\exists F.(op, q)$ positively occurs at the conclusion only if positively occurs at the ontology $\mathcal{O}$ of the side-condition. No saturation rules concern any negative occurrences. □

Theorem 21 (**Completeness for Saturation Algorithm**). *Let (D, O_-, O_+) be a (functional) safe NDO and $\mathcal{O}$ an ontology containing axioms in normal form and in $\mathcal{EL}^{\bot}(D, O_-, O_+)$. For every A, $B \in N_C^{\top} \cap sig(\mathcal{O})$, if $\mathcal{O} \models A \sqsubseteq B$, then $A \sqsubseteq B \in \mathcal{O}'$ or $A \sqsubseteq \bot \in \mathcal{O}'$.*

Proof. Initially, we are going to construct an interpretation $\mathcal{I}$ based on the axioms that $\mathcal{O}'$ contains. Subsequently, we prove that the interpretation $\mathcal{I}$ is a model of $\mathcal{O}$ for axioms of the form NF1-NF8. We prove that if $\mathcal{O} \models A \sqsubseteq B$, then $A \sqsubseteq B \in \mathcal{O}'$ or $A \sqsubseteq \bot \in \mathcal{O}'$ by showing the contrapositive. We assume that $A \sqsubseteq B \notin \mathcal{O}'$ and $A \sqsubseteq \bot \notin \mathcal{O}'$ and we show that for the model $\mathcal{I}$ we constructed (for which $\mathcal{I} \models \mathcal{O}$ is shown) it is $\mathcal{I} \not\models A \sqsubseteq B$. We call the model $\mathcal{I}$ *canonical model*.

We define the domain of the interpretation by adding to it one element for each name concept of $N_C^{\top} \cap sig(\mathcal{O})$ under the condition that the name concept is not subsumed by $\bot$:

$$\Delta^{\mathcal{I}} = \{x_A \mid A \in N_C^{\top} \cap sig(\mathcal{O}) \wedge A \sqsubseteq \bot \notin \mathcal{O}'\} \tag{4.1}$$

We continue with the interpretation of the concept names:

$$B^{\mathcal{I}} = \{x_A \mid x_A \in \Delta^{\mathcal{I}} \wedge A \sqsubseteq B \in \mathcal{O}'\} \tag{4.2}$$

Furthermore, we define the interpretation of the role names:

$$r^{\mathcal{I}} = \{(x_A, x_B) \mid A \sqsubseteq \exists r.B \in \mathcal{O}' \wedge x_A, x_B \in \Delta^{\mathcal{I}}\} \tag{4.3}$$

Now we deal with the definition of the interpretation for feature names. Before that, we define S^+ and S^-; they are parameterised with a concept name A such that $A \sqsubseteq \bot \notin \mathcal{O}'$ and a feature name each, because they are different for different axioms of the form $A \sqsubseteq \exists F.(op, q)$ (NF5) and $\exists F.(op', r) \sqsubseteq B$ (NF6):

$$S^+(A,F) = \{(op,q) \mid A \sqsubseteq \exists F.(op,q) \in \mathcal{O}'\} \tag{4.4}$$

According to Lemma 6, concepts of the form $\exists F.(op,q)$ positively occur in $\mathcal{O}'$ only if they positively occur in $\mathcal{O}$ and, as a consequence, $op \in O_+$. S^- is defined depending on A and F as follows:

$$S^-(A,F) = \{(op',r) \mid \exists F.(op',r) \sqsubseteq B \in \mathcal{O} \wedge A \sqsubseteq B \notin \mathcal{O}'\} \tag{4.5}$$

$S^+(A,F)$ and $S^-(A,F)$ are defined for the same A and F (from now on we omit the parameters of S^+ and S^- for readibility reasons).

We now show that (S^+,S^-) is indeed a constraint. For every $r \in S^+$ it holds that $r \nrightarrow_D \bot$; otherwise from rule $\mathsf{ID1}$ it would be $A \sqsubseteq \bot \in \mathcal{O}'$ which is opposite to what we have assumed. It is also true that for $(op,q) \in S^+$ and $(op',r) \in S^-$ we have $(op,q) \nrightarrow_D (op',r)$; otherwise from $A \sqsubseteq \exists F.(op,q) \in \mathcal{O}'$, $\exists F.(op',r) \sqsubseteq B \in \mathcal{O}$, $(op,q) \rightarrow_D (op',r)$ and rule $\mathsf{CR1}$ we would have $A \sqsubseteq B \in \mathcal{O}'$, which is opposite to 4.5. Additionally, since from Lemma 6 negative occurrences of axioms of the form $\exists F.(op,q)$ are preserved, it can be derived that for every op' from S^-, we have $op' \in O^-$.

For the definition of $F^{\mathcal{I}}$, we distinguish cases according to whether $\mathsf{Funct}(F) \in \mathcal{O}$ or not:

- $\mathsf{Funct}(F) \notin \mathcal{O}$

Since (S^+,S^-) is a constraint w.r.t. (D,O_-,O_+) and (D,O_-,O_+) is safe, (S^+,S^-) is satisfiable. Thus, there exists a set $V \subseteq D$ such that every $r^+ \in S^+$ is satisfied by at least one $v \in V$ but no $r^- \in S^-$ is satisfied by any $v \in V$. V depends on A and F since they uniquely define S^+ and S^-. We define $F^{\mathcal{I}}$ as:

$$F^{\mathcal{I}} = \{(x_A,v) \mid v \in V(A,F)\} \tag{4.6}$$

- $\mathsf{Funct}(F) \in \mathcal{O}$

If $\mathsf{Funct}(F) \in \mathcal{O}$, (S^+,S^-) is a functional constraint. We prove that it is a functional constraint by showing that no two datatype restrictions in S^+ are inconsistent w.r.t. D; if this was the case then from $A \sqsubseteq \exists F.(op,q)$, $A \sqsubseteq \exists F.(op',r)$, $(op,q) \wedge (op',r) \rightarrow_D \bot$ and rule $\mathsf{CD3}$, we would have $A \sqsubseteq \bot \in \mathcal{O}'$ which comes in contrast with our assumption that $A \sqsubseteq \bot \notin \mathcal{O}'$.

Since (S^+,S^-) is a functional constraint w.r.t. (D,O_-,O_+) and (D,O_-,O_+) is functionally safe, (S^+,S^-) is satisfiable. Thus, there exists a set $V \subseteq D$, with $|V| \leq 1$, such that every $r^+ \in S^+$ is satisfied by $v \in V$ but no $r^- \in S^-$ is satisfied by any $v \in V$. V depends on A and F since they uniquely define S^+ and S^-. We define $F^{\mathcal{I}}$ as:

$$F^{\mathcal{I}} = \{(x_A, v) \mid v \in V(A, F)\} \tag{4.7}$$

Since $|V| \leq 1$ the above relation is indeed a functional relation.

Now, we prove that $\mathcal{I} \models \mathcal{O}$. In order to do that, we need to prove that $\mathcal{I} \models \alpha$ for every $\alpha \in \mathcal{O}$. However, $\mathcal{O}$ contains only axioms in normal form. Therefore, it is sufficient to prove that $\mathcal{I} \models \alpha$, when α takes one of the NF1-NF8.

NF1 $A \sqsubseteq B$

We need to prove $\mathcal{I} \models A \sqsubseteq B$; in other words $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. Take an $x \in A^{\mathcal{I}}$; we show that $x \in B^{\mathcal{I}}$. By (4.2), $x = x_C$ such that $C \sqsubseteq A \in \mathcal{O}'$. Since $A \sqsubseteq B \in \mathcal{O}$ and $\mathcal{O}'$ is closed under CR1, we have $C \sqsubseteq B \in \mathcal{O}'$. Hence $x = x_C \in B^{\mathcal{I}}$ by (4.2).

We examine separately the case when $A = \top$. We have that $x_A \in \Delta^{\mathcal{I}}$ and we need to show that $x_A \in B^{\mathcal{I}}$. From rule IR2, we have that $A \sqsubseteq \top \in \mathcal{O}'$. From rule CR1, $A \sqsubseteq B \in \mathcal{O}'$ and as a consequence $x_A \in B^{\mathcal{I}}$.

NF2 $A_1 \sqcap A_2 \sqsubseteq B$

In order to prove that $\mathcal{I} \models A_1 \sqcap A_2 \sqsubseteq B$ we assume that $x \in (A_1 \sqcap A_2)^{\mathcal{I}}$ and we prove that $x \in B^{\mathcal{I}}$. Since $x \in (A_1 \sqcap A_2)^{\mathcal{I}}$, we have $x \in A_1^{\mathcal{I}}$, $x \in A_2^{\mathcal{I}}$ and $x = x_A$ for some concept name A. From $x \in A_1^{\mathcal{I}}$, $x \in A_2^{\mathcal{I}}$ and (4.2) we have $A \sqsubseteq A_1 \in \mathcal{O}'$ and $A \sqsubseteq A_2 \in \mathcal{O}'$. Since $A \sqsubseteq A_1 \in \mathcal{O}'$, $A \sqsubseteq A_2 \in \mathcal{O}'$ and $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{O}$ the axiom $A \sqsubseteq B$ has also been added to $\mathcal{O}'$ (due to rule CR2) and, therefore, $x \in B^{\mathcal{I}}$, from (4.2).

NF3 $A \sqsubseteq \exists r.B$

We show that $\mathcal{I} \models A \sqsubseteq \exists r.B$ by proving $A^{\mathcal{I}} \subseteq (\exists r.B)^{\mathcal{I}}$. Take an $x \in A^{\mathcal{I}}$; we prove that $x \in (\exists r.B)^{\mathcal{I}}$. From (4.2), $x = x_C$ where $C \sqsubseteq A \in \mathcal{O}'$. Since $A \sqsubseteq \exists r.B \in \mathcal{O}$ and $\mathcal{O}'$ is closed under CR3, we have $C \sqsubseteq \exists r.B \in \mathcal{O}'$. Since $x_C \in \Delta^{\mathcal{I}}$, we have $C \sqsubseteq \bot \notin \mathcal{O}'$ and, hence, $B \sqsubseteq \bot \notin \mathcal{O}'$ by CR5. Thus, $x_B \in \Delta^{\mathcal{I}}$ and $(x_C, x_B) \in r^{\mathcal{I}}$ by (4.3). Since $B \sqsubseteq B \in \mathcal{O}'$ by IR1, we have $x_B \in B^{\mathcal{I}}$ by (4.2). Thus, $x = x_C \in (\exists r.B)^{\mathcal{I}}$.

NF4 $\exists r.B \sqsubseteq A$

We prove that $\mathcal{I} \models \exists r.B \sqsubseteq A$ by taking an $x \in (\exists r.B)^{\mathcal{I}}$ and proving that $x \in A^{\mathcal{I}}$. Since $x \in (\exists r.B)^{\mathcal{I}}$ there exists $y \in \Delta^{\mathcal{I}}$ such that $(x, y) \in r^{\mathcal{I}}$ and $y \in B^{\mathcal{I}}$. From (4.3) and (4.2) $x = x_C$ and $y = x_D$ such that $C \sqsubseteq \exists r.D \in \mathcal{O}'$ and $D \sqsubseteq B \in \mathcal{O}'$. Since $\exists r.B \sqsubseteq A \in \mathcal{O}$ and $\mathcal{O}'$ is closed under CR4, we have $C \sqsubseteq A \in \mathcal{O}'$. Hence, $x = x_C \in A^{\mathcal{I}}$ by (4.2).

NF5 $A \sqsubseteq \exists F.(op, q)$

We prove that $\mathcal{I} \models A \sqsubseteq \exists F.(op, q)$, that is $A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}}$, by taking an $x \in A^{\mathcal{I}}$ and proving that $x \in (\exists F.(op, q))^{\mathcal{I}}$. Since $x \in A^{\mathcal{I}}$ by (4.2), there exists C such that $x = x_C$ and $C \sqsubseteq A \in \mathcal{O}'$. Since $A \sqsubseteq \exists F.(op, q) \in \mathcal{O}$ and $\mathcal{O}'$ is closed under CD2, we have $C \sqsubseteq \exists F.(op, q) \in \mathcal{O}'$. Let (S^+, S^-) be the constraint constructed for F and C as defined in (4.4) and (4.5). Then $(op, q) \in S^+$. By (4.6) we have $(x_C, v) \in F^{\mathcal{I}}$ for every $v \in V$, where V is a solution for (S^+, S^-). Since V is a solution for (S^+, S^-) and $(op, q) \in S^+$, there exists $v \in V$ such that v satisfies (op, q). Hence, $x = x_C \in (\exists F.(op, q))^{\mathcal{I}}$.

NF6 $\exists F.(op', r) \sqsubseteq B$

We prove that $\mathcal{I} \models \exists F.(op, q) \sqsubseteq B$, that is $(\exists F.(op, q))^{\mathcal{I}} \subseteq B^{\mathcal{I}}$. Take an arbitrary $x \in (\exists F.(op, q))^{\mathcal{I}}$. We have to show that $x \in B^{\mathcal{I}}$. We have that $x = x_C$ for some concept name C. Let (S^+, S^-) be the constraint for F and C. Since $x_C \in (\exists F.(op', r))^{\mathcal{I}}$, by (4.6), there exists $v \in V$, such that v satisfies (op', r), where V is a solution for (S^+, S^-). Hence, $(op', r) \notin S^-$, and so, $C \sqsubseteq B \in \mathcal{O}'$ by definition of S^-. Now from (4.2) we have that $x_C \in B^{\mathcal{I}}$.

NF7 $A \sqsubseteq \bot$

We need to prove that $A^{\mathcal{I}} = \emptyset$. Assume to the contrary that there exists $x \in A^{\mathcal{I}}$. Then by (4.2) $x = x_C$ for some concept name C such that $C \sqsubseteq A \in \mathcal{O}'$. Since $\mathcal{O}'$ is closed under CR1 and $A \sqsubseteq \bot \in \mathcal{O}'$, we have $C \sqsubseteq \bot \in \mathcal{O}'$. Hence, $x_C \notin \Delta^{\mathcal{I}}$ by (4.1).

NF8 $\mathsf{Funct}(F)$

We assume that there exists $x \in \Delta^{\mathcal{I}}$, such that $(x, v_1), (x, v_2) \in F^{\mathcal{I}}$. According to the semantics of $\mathsf{Funct}(F)$, it is sufficient to prove that $v_1 = v_2$. From (4.7), V can contain at most one element and, consequently, $v_1 = v_2$.

We now show that if $A \sqsubseteq B \notin \mathcal{O}'$ and $A \sqsubseteq \bot \notin \mathcal{O}'$, then there is a model $\mathcal{I}$ of $\mathcal{O}$ such that $\mathcal{I} \not\models A \sqsubseteq B$. We choose as a model the canonical model $\mathcal{I}$. $A^{\mathcal{I}} \not\subseteq B^{\mathcal{I}}$ holds, since $x_A \in \Delta^{\mathcal{I}}$ ($A \sqsubseteq \bot \notin \mathcal{O}'$) and $x_A \in A^{\mathcal{I}}$ (using rule IR1) but $x_A \notin B^{\mathcal{I}}$ since $A \sqsubseteq B \notin \mathcal{O}'$.

□

4.3 Non-convexity and Intractability

The current section deals with the non-convex NDOs and proves that the subsumption problem in $\mathcal{EL}^{\bot}(D, O_-, O_+)$, where (D, O_-, O_+) is a non-convex NDO, is EXPTIME-hard.

Lemma 7 (**Non-convexity and Intractability Lemma**). *For every non-convex NDO* (D, O_-, O_+) *and every ontology* $\mathcal{O}$ *expressed in* $\mathcal{ELU}$*, there can be constructed in polynomial time an ontology* $\mathcal{O}'$ *in* $\mathcal{EL}^{\bot}(D, O_-, O_+)$*, such that* ***(1)*** *and* ***(2)*** *are equivalent:*

(1) $\mathcal{O} \models \alpha$

(2) $\mathcal{O}' \models \alpha$

on condition that α *does not contain any concept of the form* $\exists F.(op, q)$.

Proof. Since (D, O_-, O_+) is a non-convex NDO and from the definition of convexity there are $op \in O_+$ and op_1, $op_2 \in O_-$, such that the following three conditions hold (we take w.l.o.g. the minimum requirement for violation of convexity):

1. $(op, q) \rightarrow_D (op_1, r_1) \vee (op_2, r_2)$
2. $(op, q) \nrightarrow_D (op_1, r_1)$
3. $(op, q) \nrightarrow_D (op_2, r_2)$

First, we normalize the ontology $\mathcal{O}$ in order to make disjunction occur only in axioms of the form $A \sqsubseteq B \sqcup C$. To achieve that we apply Algorithm 1 to $\mathcal{O}$ augmented by the following two rules:

NR	head	body
NR6	$B \sqsubseteq G \sqcup C$	$\{G \sqsubseteq A, B \sqsubseteq A \sqcup C\}$
NR7	$B \sqcup C \sqsubseteq D$	$\{B \sqsubseteq C, B \sqsubseteq D\}$

where A is a fresh concept w.r.t. the, so far, tranformed ontology, $G \notin N_C^{\top}$, $D \in N_C^{\top}$ name and B, C and G are concepts. Rule NR6 is applied modulo commutativity of disjunction. If $\mathcal{O}'_0$ is the resulting ontology, for an axiom α that does not contain any of the introduced fresh concepts it can be proved that $\mathcal{O} \models \alpha \Leftrightarrow \mathcal{O}'_O \models \alpha$, using an extended version of the correctness proof for the Normalization Algorithm (page 23).

Now, we deal with axioms of the form $A \sqsubseteq B \sqcup C$. Suppose that $\mathcal{O}'_0$ contains n axioms of the form $A \sqsubseteq B \sqcup C$, where $A, B, C \in N_C$. We set $\mathcal{O}_0 = \mathcal{O}'_0$ and the ontologies $\mathcal{O}_1, \ldots, \mathcal{O}_n$ are the resulting ontologies of the following transformation for each one of the n axioms:

$$\mathcal{O}_{i+1} = (\mathcal{O}_i \setminus \{A \sqsubseteq B \sqcup C\}) \cup$$
$$\{A \sqsubseteq \exists F.(op, q),\ \exists F.(op_1, r_1) \sqsubseteq B,\ \exists F.(op_2, r_2) \sqsubseteq C\} \tag{4.8}$$

where $F \in N_F$ is a fresh feature name. We set $\mathcal{O}' = \mathcal{O}_n$. We can show in a similar way as in the complexity proof of Algorithm 1 that the above transformation from $\mathcal{O}$ to $\mathcal{O}'$ has polynomial cost.

(1) ⇒ (2): $\mathcal{O} \models \alpha \Rightarrow \mathcal{O}' \models \alpha$, for every axiom α.

We show it by initially proving that $\mathcal{O}_i \models \alpha \Rightarrow \mathcal{O}_{i+1} \models \alpha$. Then, if we combine the implications for $i = 0, \ldots, n-1$ we obtain $\mathcal{O}_0 \models \alpha \Rightarrow \mathcal{O}_n \models \alpha$, which is $\mathcal{O} \models \alpha \Rightarrow \mathcal{O}' \models \alpha$. We assume that:

$$\mathcal{O}_i \models \alpha \tag{4.9}$$

and we need to prove that $\mathcal{O}_{i+1} \models \alpha$. From (4.9) we have that for every interpretation $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{O}_i \Rightarrow \mathcal{I} \models \alpha \tag{4.10}$$

In order to prove $\mathcal{O}_{i+1} \models \alpha$, we assume that:

$$\mathcal{I} \models \mathcal{O}_{i+1} \tag{4.11}$$

and we prove $\mathcal{I} \models \alpha$. By (4.11) and (4.8) we have that:

$$A^{\mathcal{I}} \subseteq (\exists F.(op, q))^{\mathcal{I}} \tag{4.12}$$

$$(\exists F.(op_1, r_1))^{\mathcal{I}} \subseteq B^{\mathcal{I}} \tag{4.13}$$

$$(\exists F.(op_2, r_2))^{\mathcal{I}} \subseteq C^{\mathcal{I}} \tag{4.14}$$

We prove that $\mathcal{I} \models \mathcal{O}_i$, that is $\mathcal{I} \models A \sqsubseteq B \sqcup C$ and, thus, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}} \cup C^{\mathcal{I}}$. Take a $x \in A^{\mathcal{I}}$. From (4.12) we have that $x \in (\exists F.(op, q))^{\mathcal{I}}$, that is, there is an r such that $(x, r) \in F^{\mathcal{I}}$ and r satisfies (op, q). However, since $(op, q) \rightarrow_D (op_1, r_1) \vee (op_2, r_2)$, r also satisfies (op_1, r_1) or (op_2, r_2). So, we have $x \in (\exists F.(op_1, q_1))^{\mathcal{I}}$ or $x \in (\exists F.(op_2, q_2))^{\mathcal{I}}$. Therefore, from (4.13) and (4.14) $x \in B^{\mathcal{I}}$ or $x \in C^{\mathcal{I}}$, that is $x \in (B \cup C)^{\mathcal{I}}$, which was required to show. Therefore, we have shown that $\mathcal{I} \models \mathcal{O}_i$ and from (4.10) we have that $\mathcal{I} \models \alpha$ which we needed to show.

(2) ⇒ (1): $\mathcal{O}' \models \alpha \Rightarrow \mathcal{O} \models \alpha$, for every axiom α that does not contain any concept of the form $\exists F.(op, q)$.

Again, we initially prove that $\mathcal{O}_{i+1} \models \alpha \Rightarrow \mathcal{O}_i \models \alpha$. Then, by combining the implications for $i = n-1, \ldots, 0$ we have that $\mathcal{O}_n \models \alpha \Rightarrow \mathcal{O}_0 \models \alpha$, which is the same as $\mathcal{O}' \models \alpha \Rightarrow \mathcal{O} \models \alpha$. We assume that $\mathcal{O}_{i+1} \models \alpha$, that is for every interpretation $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{O}_{i+1} \Rightarrow \mathcal{I} \models \alpha \tag{4.15}$$

In order to prove that $\mathcal{O}_i \models \alpha$, we prove that for every interpretation $\mathcal{I}$ if:

$$\mathcal{I} \models \mathcal{O}_i \tag{4.16}$$

then $\mathcal{I} \models \alpha$.

We define an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{O}_{i+1}$ and $\mathcal{I}|_{sig(\mathcal{O}_i)} = \mathcal{J}|_{sig(\mathcal{O}_i)}$.

$$X^{\mathcal{J}} := X^{\mathcal{I}} \text{ if } X \in N_C \uplus N_R \uplus N_F \setminus \{F\} \tag{4.17}$$

$$F^{\mathcal{J}} := \begin{cases} \{(x,b)\} & \text{if } x \in B^{\mathcal{I}} \cap A^{\mathcal{I}}, \\ \{(x,c)\} & \text{if } x \in (C^{\mathcal{I}} \setminus B^{\mathcal{I}}) \cap A^{\mathcal{I}}, \\ \emptyset & \text{otherwise.} \end{cases} \tag{4.18}$$

where $b, c \in D$. We choose number b in such a way that it satisfies (op, q) and (op_1, r_1) but not (op_2, r_2), which is possible since $(op, q) \nrightarrow_D (op_2, r_2)$. Similarly, we choose number c in such a way that it satisfies (op, q) and (op_2, r_2) but not (op_1, r_1), which is possible since $(op, q) \nrightarrow_D (op_1, r_1)$.

We need to prove that:

- $A^{\mathcal{J}} \subseteq (\exists F.(op, q))^{\mathcal{J}}$
  Take an arbitrary $x \in A^{\mathcal{J}}$. Since $\mathcal{I}|_{sig(\mathcal{O}_i)} = \mathcal{J}|_{sig(\mathcal{O}_i)}$, then $A^{\mathcal{J}} = A^{\mathcal{I}}$ and, thus, $x \in A^{\mathcal{I}}$. From (4.16) and $A \sqsubseteq B \sqcup C \in \mathcal{O}_i$ we have that $A^{\mathcal{I}} \subseteq B^{\mathcal{I}} \cup C^{\mathcal{I}}$. Therefore, $x \in B^{\mathcal{I}}$ or $x \in C^{\mathcal{I}}$. If $x \in B^{\mathcal{I}}$, then $x \in B^{\mathcal{I}} \cap A^{\mathcal{I}}$ as well and, so, $(x, b) \in F^{\mathcal{J}}$ and $x \in (\exists F.(op, q))^{\mathcal{J}}$ (since b satisfies (op, q)). Otherwise $x \in (C^{\mathcal{I}} \setminus B^{\mathcal{I}}) \cap A^{\mathcal{I}}$ and so, $(x, c) \in F^{\mathcal{J}}$ and $x \in (\exists F.(op, q))^{\mathcal{J}}$ (since c satisfies (op, q)).
- $(\exists F.(op_1, r_1))^{\mathcal{J}} \subseteq B^{\mathcal{J}}$
  Take an arbitrary $x \in (\exists F.(op_1, r_1))^{\mathcal{J}}$. In that case, from the definition of $F^{\mathcal{J}}$, $(x, b) \in F^{\mathcal{J}}$, because $(x, c) \in F^{\mathcal{J}}$ cannot be the case since c does not satisfy (op_1, q_1). Consequently, $x \in B^{\mathcal{I}} \cap A^{\mathcal{I}}$ and, thus, $x \in B^{\mathcal{I}}$ which was required to show.
- $(\exists F.(op_2, r_2))^{\mathcal{J}} \subseteq C^{\mathcal{J}}$
  Take an arbitrary $x \in (\exists F.(op_2, r_2))^{\mathcal{J}}$. In that case, from the definition of $F^{\mathcal{J}}$, $(x, c) \in F^{\mathcal{J}}$ (because $(x, b) \in F^{\mathcal{J}}$ cannot be the case since b does not satisfy (op_2, q_2)) and, thus, $x \in C^{\mathcal{I}} \cap A^{\mathcal{I}}$. As a consequence, $x \in C^{\mathcal{I}}$ and, thus, $x \in (C^{\mathcal{I}} \setminus B^{\mathcal{I}}) \cap A^{\mathcal{I}}$ which we needed to prove.

From the above we have that $\mathcal{J} \models \mathcal{O}_{i+1}$ and from (4.15) we deduce that:

$$\mathcal{J} \models \alpha \tag{4.19}$$

Since α does not contain any fresh concepts w.r.t. $\mathcal{O}_i$ and $\mathcal{I}|_{sig(\mathcal{O}_i)} = \mathcal{J}|_{sig(\mathcal{O}_i)}$, we have that $\mathcal{I} \models \alpha$, which was required to show.

□

Example 12. *The goal of this example is to show the necessity of the restriction of Lemma 7 that axiom α should not contain any concept of the form $\exists F.(op, q)$. Let $\mathcal{O}$ be the initial ontology in $\mathcal{ELU}$ and $\mathcal{O}'$ the reduced ontology as follows:*

$$\mathcal{O} = \{A \sqsubseteq B \sqcup C\}$$

$$\mathcal{O}' = \{A \sqsubseteq \exists F.(op, q), \exists F.(op_1, r_1) \sqsubseteq B, \exists F.(op_2, r_2) \sqsubseteq C\}$$

We prove that there is an axiom which does not satisfy the above restriction and $\mathcal{O}' \models \alpha$ but $\mathcal{O} \not\models \alpha$. For $\alpha = A \sqsubseteq \exists F.(op, q)$, we have that $\mathcal{O}' \models \alpha$, since $\alpha \in \mathcal{O}'$. Let $\mathcal{I}$ be the following interpretation :

$$\Delta^{\mathcal{I}} = \{x\}$$

$$A^{\mathcal{I}} = \{x\}$$

$$B^{\mathcal{I}} = \{x\}$$

$$X^{\mathcal{I}} = \emptyset \text{ for } X \notin \{A, B\}$$

We have that $\mathcal{I} \models \mathcal{O}$, (because $A^{\mathcal{I}} = \{x\} \subseteq \{x\} = (B \sqcup C)^{\mathcal{I}}$), but $\mathcal{I} \not\models \alpha$ because $A^{\mathcal{I}} = \{x\} \not\subseteq \emptyset = (\exists F.(op, q))^{\mathcal{I}}$. Therefore, $\mathcal{O} \not\models \alpha$, which is a consequence of the occurrence of $\exists F.(op, q)$ in α.

Theorem 22. *In $\mathcal{EL}^{\perp}(D, O_-, O_+)$, where (D, O_-, O_+) is a non-convex NDO, the subsumption problem is EXPTIME-hard.*

Proof. The same reasoning task in $\mathcal{ELU}$ is EXPTIME-complete [1]. From Lemma 7 the subsumption problem $\mathcal{O} \models \alpha$ in $\mathcal{ELU}$ can be reduced in polynomial time to the subsumption problem in $\mathcal{EL}^{\perp}(D, O_-, O_+)$, where (D, O_-, O_+) is a non-convex NDO. Hence, the subsumption problem in $\mathcal{EL}^{\perp}(D, O_-, O_+)$, where (D, O_-, O_+) is a non-convex NDO, is EXPTIME-hard.

□

A similar result has been shown for non-convex concrete domains [1]; however, in that case the convexity property is less precisely defined and is not connected to specific NDOs. Additionally, the technique used in the proof is different from the one used here. In our case we reduce the ontology in $\mathcal{ELU}$ to an ontology in $\mathcal{EL}^{\bot}(D, O_{-}, O_{+})$, where (D, O_{-}, O_{+}) is a non-convex NDO. In the other case the ontology in $\mathcal{ELU}$ is first reduced to an ontology with a single disjunction axiom and then to an ontology in $\mathcal{EL}(\mathcal{D})$, where $\mathcal{D}$ is a non-convex concrete domain.

Chapter 5

Important NDO Instances

The present chapter points out specific instances of tractable and intractable $\mathcal{EL}^{\bot}(D)$ fragments by dealing with particular NDOs and proving which of them are safe and which not. Firstly, we define in Section 5.1 some properties that datatypes D can exhibit and are used later in the chapter. Section 5.2 presents some minimal instances of non-safe NDOs, while Section 5.3 discusses some maximal instances of safe NDOs. At last, Section 5.4 sums up the overall results and gives a general overview.

5.1 Datatypes with Special Properties

We pinpoint some special properties of datatypes D which will later be used both in safety and non-safety cases.

Definition 23 (**Density and Double-non-density**). *Let D be a datatype. We say that D is a* dense *datatype iff for every a, $b \in D$ there is a $c \in D$ such that $a < c < b$. We say that D is a* double non-dense *datatype iff there exist a, b and $c \in D$ such that $a < b < c$ and there is no $d \in D$ with $a < d < b$ or $b < d < c$.*

Definition 24 (**Previous/Next Elements and Sparsity**). *Let D be a datatype. If D is a non-dense datatype and for a, $b \in D$ with $a < b$ there is no $c \in D$ such that $a < c < b$, we say that b (a) is the next (previous) element of a (b). We also define the respective functions $next : D \to D$ and $previous : D \to D$ which return the next and previous element of an element. We call a datatype D* sparse *when the $min(D)$ (if there exists one) has the next element, the $max(D)$ (if there exists one) has the previous element and all the remaining elements of the datatype have the previous and the next element.*

Definition 25 (**Left(Right)-openness**). *Let D be a datatype. We say that D is a* right-open *(or* left-open*) datatype iff it does not have a $max(D)$ (or $min(D)$) element.*

Example 13. $\mathbb{R}^+ \cap \{0\}$ *is an example of a right-open dense datatype and* $\mathbb{Z}$ *is an example of a sparse datatype.*

5.2 Minimal Non-safe NDOs

In the current section we present cases of non-safe NDOs and prove their non-safety. The following non-safe NDOs are minimal, that is if one of the relational operators is removed from O_- or O_+ they become safe. The NDOs which are symmetric to the ones we present here are non-safe as well; the corresponding proof is symmetric to the given proof. By symmetric cases and symmetric proof we mean that the relational operators have been replaced with the operators of opposite direction (i.e., $\leq \Leftrightarrow \geq$, $< \Leftrightarrow >$ and $= \Leftrightarrow =$).

Proposition 8. $(D_2, \{\leq, \geq\}, \{\geq\})$, $(D_2, \{\leq, \geq\}, \{>\})$, $(D_2, \{\leq, >\}, \{\geq\})$, $(D_2, \{\leq, >\}, \{>\})$, $(D_3, \{<, >\}, \{\geq\})$, $(D_3, \{<, >\}, \{>\})$ *and* $(D_1, \{<, =\}, \{\leq\})$, *where* D_1, D_2 *and* D_3 *are datatypes that have at least* 1, 2 *and* 3 *elements respectively, are non-safe NDOs.*

Proof. From Proposition 5 in order to prove non-safety it suffices to prove non-convexity. From Proposition 4 it is adequate for non-convexity to provide a violation convexity condition. The following table supplies such a violation convexity condition $((op, q), (op_1, r_1), (op_2, r_2))$ for every NDO:

NDO	(op, q)	(op_1, r_1)	(op_2, r_2)	Condition
$(D_2, \{\leq, \geq\}, \{\geq\})$	$(\geq, p)$	$(\leq, q)$	$(\geq, q)$	$p < q$
$(D_2, \{\leq, \geq\}, \{>\})$	$(>, p)$	$(\leq, q)$	$(\geq, q)$	$p < q$
$(D_2, \{\leq, >\}, \{\geq\})$	$(\geq, p)$	$(\leq, q)$	$(>, q)$	$p \leq q$
$(D_2, \{\leq, >\}, \{>\})$	$(>, p)$	$(\leq, q)$	$(>, q)$	$p < q$
$(D_3, \{<, >\}, \{\geq\})$	$(\geq, p)$	$(<, r)$	$(>, q)$	$p < q < r$
$(D_3, \{<, >\}, \{>\})$	$(>, p)$	$(<, r)$	$(>, q)$	$p < q < r$
$(D_1, \{<, =\}, \{\leq\})$	$(\leq, p)$	$(<, p)$	$(=, p)$	-

□

Proposition 9. $(D_{nd}, \{\leq, =\}, \{\leq\})$, $(D_{nd}, \{<, =\}, \{<\})$, $(D_{2nd}, \{\leq, =\}, \{<\})$, *where* D_{nd} *is a non-dense datatype and* D_{2nd} *is a double non-dense datatype, are non-safe NDOs.*

Proof. Similarly to the previous case we only need the violation convexity conditions $((op, q), (op_1, r_1), (op_2, r_2))$ which are given in the following table:

NDO	(op, q)	(op_1, r_1)	(op_2, r_2)
$(D_{nd}, \{\leq, =\}, \{\leq\})$	$(\leq, p)$	$(=, p)$	$(\leq, q)$
$(D_{nd}, \{<, =\}, \{<\})$	$(<, p)$	$(<, q)$	$(=, q)$
$(D_{2nd}, \{\leq, =\}, \{<\})$	$(<, p)$	$(\leq, r)$	$(=, q)$

For the first two NDOs we require $q < p$ and $\nexists r \in D_{nd}$ such that $q < r < p$. For the third NDO we choose p, r and $q \in D_{2nd}$, such that $r < q < p$ and $\nexists s \in D_{2nd}$ such that $r < s < q$ or $q < s < p$. □

Proposition 10. *$(D_{nr}, \{\leq, =\}, \{\geq\})$, $(D_{nr}, \{\leq, =\}, \{>\})$, $(D_{nr}, \{<, =\}, \{\geq\})$ and $(D_{nr}, \{<, =\}, \{>\})$, where D_{nr} is a non-right-open datatype such that there exists a $q \in D_{nr}$ with $q < max_{D_{nr}}$ but there exists no $r \in D_{nr}$ with $q < r < max_{D_{nr}}$, are non-safe NDOs.*

Proof. We give the violation convexity conditions $((op, q), (op_1, r_1), (op_2, r_2))$:

NDO	(op, q)	(op_1, r_1)	(op_2, r_2)
$(D_{nr}, \{\leq, =\}, \{\geq\})$	$(\geq, p)$	$(=, max_{D_{nr}})$	$(\leq, q)$
$(D_{nr}, \{\leq, =\}, \{>\})$	$(>, p)$	$(=, max_{D_{nr}})$	$(\leq, q)$
$(D_{nr}, \{<, =\}, \{\geq\})$	$(\geq, p)$	$(=, max_{D_{nr}})$	$(<, max_{D_{nr}})$
$(D_{nr}, \{<, =\}, \{>\})$	$(>, p)$	$(=, max_{D_{nr}})$	$(<, max_{D_{nr}})$

For the first two NDOs we choose p and q such that $p < q$ and $\nexists r \in D_{nr}$ with $q < r < max_{D_{nr}}$.

□

5.3 Maximal Safe NDOs

Now, we present a series of safe NDOs and prove their safety. The safe NDOs of Propositions 11, 12, 13, 14, 15 and 16 are maximal, that is if one of the relational operators is added to O_- or O_+ they become non-safe. Like before, the NDOs which are symmetric to the ones we present here are safe as well.

Proposition 11. *The NDO $(D_d, \{\leq, =\}, \{<, >, \leq, \geq, =\})$, where D_d is a dense, left- and right-open datatype, is a safe NDO.*

Proof. In order to prove that the NDO is safe, it is sufficient to prove that every constraint (S^+, S^-) w.r.t. this NDO is satisfiable, that is, there is a set $V \subseteq D_d$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally, no $v \in V$ satisfies any $r \in S^-$. Let (S^+, S^-) be a constraint. We set the following values:

$$p_1 = \min\{p \mid (<, p) \in S^+\},\ p_2 = \max\{p \mid (>, p) \in S^+\}$$
$$p_3 = \min\{p \mid (\leq, p) \in S^+\},\ p_4 = \max\{p \mid (\geq, p) \in S^+\}$$
$$n = \max\{n' \mid (\leq, n') \in S^-\}$$

It is necessarily $n < \min(p_1, p_3)$, because otherwise we would have $n \geq \min(p_1, p_3)$ and, thus, $(<, p_1) \rightarrow_D (\leq, n)$ or $(\leq, p_3) \rightarrow_D (\leq, n)$ which is prohibited by the definition of S^-. We define V as:

$$V = \{v_1, v_2\} \cup \{v \mid (=, v) \in S^+\}$$

such that $n < v_1 < \min(p_1, p_3)$ (left-openness if n does not exist), $v_2 > \max(n, p_2, p_4)$ (right-openness) and v_1, $v_2 \neq q$ for $(=, q) \in S^-$. There can always be found values (for v_1 and v_2) between two other fixed values and $\neq q$ for $(=, q) \in S^-$, because the datatype D_d is dense. If one of the max or min partial functions does not return a value because the respective set is empty (e.g., there is no p such that $(<, p) \in S^+$), we relax the corresponding restriction (e.g., instead of $v_1 < \min(p_1, p_3)$ it is $v_1 < p_3$). We now show that every $r \in S^+$ is satisfied. We distinguish cases for r:

- $(<, p)$ and $(\leq, p)$ are satisfied by v_1, since $v_1 < \min(p_1, p_3)$ and $\min(p_1, p_3) \leq p$ from the definition of p_1, p_3
- $(>, p)$ and $(\geq, p)$ is satisfied by v_2, since $v_2 > \max(n, p_2, p_4)$ and $\max(p_2, p_4) \geq p$ from the definition of p_2, p_4
- $(=, p)$ is satisfied by p which by definition has been added to V

On the other hand, no $r \in S^-$ is satisfied because:

- $v_1 > n$ and, thus, v_1 does not satisfy any $(\leq, n') \in S^-$ with $n' \leq n$
- $v_2 > \max(n, p_2, p_4) \geq n$ and, thus, v_2 does not satisfy any $(\leq, n') \in S^-$ because $n' \leq n$
- v_1, $v_2 \neq q$ for every $(=, q) \in S^-$

□

Proposition 12. *The NDO $(D_d, \{<, \leq\}, \{<, >, \leq, \geq, =\})$, where D_d is a dense, left- and right-open datatype, is a safe NDO.*

Proof. As previously, we need to prove that every constraint (S^+, S^-) w.r.t. this NDO is satisfiable, that is, there is a set $V \subseteq D_d$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally, no $v \in V$ satisfies any $r \in S^-$. Let (S^+, S^-) be a constraint. We set the following values:

$$p_1 = \min\{p \mid (<, p) \in S^+\},\ p_2 = \max\{p \mid (>, p) \in S^+\}$$
$$p_3 = \min\{p \mid (\leq, p) \in S^+\},\ p_4 = \max\{p \mid (\geq, p) \in S^+\}$$
$$n_1 = \max\{n \mid (<, n) \in S^-\},\ n_2 = \max\{n \mid (\leq, n) \in S^-\}$$

We define V as:

$$V = \{v_1, v_2\} \cup \{v \mid (=, v) \in S^+\}$$

where $v_2 > \max(n_1, n_2, p_2, p_4)$ (right-openness) and we distinguish two cases for the definition of v_1:

$p_1 \leq p_3$ We take v_1 such that $\max(n_1, n_2) < v_1 < p_1$ (which is possible since D_d is a dense and left-open datatype; left-openness is required if $\max(n_1, n_2)$ does not exist).

$p_3 < p_1$ We take $v_1 = p_3$.

As previously, if one of the max or min partial functions does not return a value because the respective set is empty, we relax the corresponding restriction. Specifically, if there are no $(\leq, p) \in S^+$ we take the first case above and if there are no $(<, p) \in S^+$ we take the second case above. We now show that every $r \in S^+$ is satisfied. We distinguish cases for r:

- $(<, p)$ is satisfied either by $v_1 < p_1$ or by $v_1 = p_3 < p_1$, where $p_1 \leq p$
- $(>, p)$ and $(\geq, p)$ are satisfied by v_2, since $v_2 > \max(n_1, n_2, p_2, p_4)$
- $(\leq, p)$ is satisfied either by $v_1 < p_1 \leq p_3$ or by $v_1 = p_3$, where $p_3 \leq p$
- $(=, p)$ is satisfied by p which by definition has been added to V

At the same time, no $r \in S^-$ is satisfied because:

- either $v_1 > \max(n_1, n_2)$ and, thus, v_1 does not satisfy $(<, n_1)$ or $v_1 = p_3$ and, still v_1 does not satisfy $(<, n_1)$ because $n_1 \leq p_3$. $n_1 \leq p_3$ is a consequence of $(\leq, p_3) \nrightarrow_{D_d} (<, n_1)$ since $(<, n_1) \in S^-$.
- either $v_1 > \max(n_1, n_2)$ and, thus, v_1 does not satisfy $(\leq, n_2)$ or $v_1 = p_3$ and, still v_1 does not satisfy $(\leq, n_2)$ because $n_2 < p_3$. $n_2 < p_3$ is a consequence of $(<, p_3) \nrightarrow_{D_d} (\leq, n_2)$ because $(\leq, n_2) \in S^-$.
- $v_2 > \max(n_1, n_2, p_2, p_4) > n$ and, thus, v_2 does not satisfy neither $(<, n_1)$ nor $(\leq, n_2)$

□

Proposition 13. *The NDO $(D_d, \{<, \leq, =\}, \{<, >, \geq, =\})$, where D_d is a dense, left- and right-open datatype, is a safe NDO.*

Proof. As done before, we need to prove that every constraint (S^+, S^-) w.r.t. this NDO is satisfiable, that is, there is a set $V \subseteq D_d$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally, no $v \in V$ satisfies any $r \in S^-$. Let (S^+, S^-) be a constraint. We set:

$$p_1 = \min\{p \mid (<, p) \in S^+\},\ p_2 = \max\{p \mid (>, p) \in S^+\}$$
$$p_3 = \max\{p \mid (\geq, p) \in S^+\},\ n_1 = \max\{n \mid (<, n) \in S^-\}$$
$$n_2 = \max\{n \mid (\leq, n) \in S^-\}$$

It is necessarily $n_1, n_2 < p_1$, because if $p_1 \leq n_1$ (or $p_1 \leq n_2$), then $(<, p_1) \rightarrow_D (<, n_1)$ (or $(<, p_1) \rightarrow_D (\leq, n_2)$) and $(<, n_1)$ (or $(<, n_2)$) would not be in S^-. We define V as:

$$V = \{v_1, v_2\} \cup \{v \mid (=, v) \in S^+\}$$

such that $\max(n_1, n_2) < v_1 < p_1$ (which is possible since D_d is a dense and left-open datatype; left-openness is required if $\max(n_1, n_2)$ does not exist), $v_2 > \max(n_1, n_2, p_2, p_3)$ (since D_d is a right left-open datatype) and v_1, $v_2 \neq q$ for $(=, q) \in S^-$. Every $r \in S^+$ is satisfied because:

- $(<, p)$ is satisfied by v_1, since $v_1 < p_1 \leq p$
- $(>, p)$ and $(\geq, p)$ are satisfied by v_2, since $v_2 > \max(n_1, n_2, p_2, p_3) \geq p$
- $(=, p)$ is satisfied by p which by definition has been added to V

On the other hand, no $r \in S^-$ is satisfied because:

- $v_1 > \max(n_1, n_2)$ and, thus, v_1 neither satisfies $(<, n_1)$ nor $(\leq, n_2)$
- $v_2 > \max(n_1, n_2, p_2, p_3)$ and, thus, v_2 neither satisfies $(<, n_1)$ nor $(\leq, n_2)$
- v_1, $v_2 \neq q$ for every $(=, q) \in S^-$

□

Proposition 14. *The NDO $(D_s, \{<, \leq\}, \{<, \leq, >, \geq, =\})$, where D_s is a sparse, left- and right-open datatype, is a safe NDO.*

Proof. As in the previous cases, we need to prove that every constraint (S^+, S^-) w.r.t. this NDO is satisfiable, that is, there is a set $V \subseteq D_s$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally, no $v \in V$ satisfies any $r \in S^-$. Let (S^+, S^-) be a constraint. We can replace every $(\leq, p) \in S^-$, $(\leq, p) \in S^+$ and $(\geq, p) \in S^+$ with $(<, p')$, $(<, p')$ and $(>, p'')$ respectively, where p' is the next element of p and p'' is the previous element of p. Since D_s is a left- and right-open datatype the next and previous element of a $p \in D_s$ always exist. We set:

$$p_1 = \min\{p \mid (<, p) \in S^+\}$$
$$p_2 = \max\{p \mid (>, p) \in S^+\}$$
$$n = \max\{n' \mid (<, n') \in S^-\}$$

It is necessarily $n < p_1$, because if $p_1 \leq n$, it would be $(<, p_1) \rightarrow_D (<, n)$ and so $(<, n) \notin S^-$. We distinguish two cases for the definition of V:

- $p_1 \leq next(p_2)$

$$V = \{v_1, v_2\} \cup \{v \mid (=, v) \in S^+\}$$

  where $v_1 = previous(p_1)$ and $v_2 = next(p_2)$.

- $next(p_2) < p_1$

$$V = \{v_1\} \cup \{v \mid (=, v) \in S^+\}$$

  where $v_1 = previous(p_1)$.

Every $r \in S^+$ is satisfied because:

- $(<, p)$ is satisfied by v_1, since $v_1 = previous(p_1)$ and $p_1 \leq p$
- $(>, p_2)$ is either satisfied by $v_2 = next(p_2) > p_2$ when $p_1 \leq next(p_2)$ or by $v_1 = previous(p_1) > p_2$ when $p_1 > next(p_2)$
- $(=, p)$ is satisfied by p which by definition has been added to V.

Moreover, no $r \in S^-$ is satisfied because:

- $v_1 = previous(p_1) \geq n$ and, thus, v_1 does not satisfy any $(<, n')$ with $n' \leq n$
- $v_2 = next(p_2) \geq p_1 > n$ and, thus, v_2 does not satisfy $(<, n')$ with $n' \leq n$

□

Proposition 15. *The NDO $(D_s, \{<, \leq, =\}, \{>, \geq, =\})$, where D_s is a sparse, left- and right-open datatype, is a safe NDO.*

Proof. Like before, we need to prove that every constraint (S^+, S^-) w.r.t. this NDO is satisfiable, that is, there is a set $V \subseteq D_s$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally, no $v \in V$ satisfies any $r \in S^-$. Let (S^+, S^-) be a constraint. We can replace every $(\leq, p) \in S^-$ and $(\geq, p) \in S^+$ with $(<, p')$ and $(>, p'')$ correspondingly; p' is the next element of p and p'' is the previous element of p. Since D_s is a left- and right-open datatype the next and previous element of a $p \in D_s$ always exist. Thus, we end up with an S^+ which only contains datatype restrictions of the form $(<, p)$ and $(=, p)$ and an S^- with datatype restrictions of the form $(>, p)$ and $(=, p)$. We define V:

$$V = \{v_1\} \cup \{v \mid (=, v) \in S^+\}$$

such that $v_1 > \max(n, p_1)$ (which is possible since D_s is a right-open datatype) and $v_1 \neq q$, for every $(=, q) \in S^-$, where:

$$p_1 = \max\{p \mid (>, p) \in S^+\},\ n = \max\{n' \mid (<, n') \in S^-\}$$

Every $r \in S^+$ is satisfied because:

- $(>, p)$ is satisfied by $v_1 > \max(n, p_1) \geq p$
- $(=, p)$ is satisfied by p which by definition has been added to V.

Additionally, no $r \in S^-$ is satisfied because:

- $v_1 > \max(n, p_1) \geq n'$ and, thus, v_1 does not satisfy $(<, n')$
- $v_1 \neq q$, for every $(=, q) \in S^-$

□

Proposition 16. *The NDO* $(D, \{<, \leq, >, \geq, =\}, \{=\})$ *is safe.*

Proof. As done previously, we need to prove that there is a set $V \subseteq D$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally there exists no $r \in S^-$, which is satisfied by any $v \in V$. We define V as:

$$V = \{v \mid (=, v) \in S^+\}$$

Every $r \in S^+$ is satisfied because of the definition. Moreover, no $r \in S^-$ is satisfied because no $v \in V$ satisfies any of the datatype restrictions of S^-; in that case, we would have $(=, v) \rightarrow_D r^-$ for some $r^- \in S^-$, which is prohibited by the definition of S^-.

□

Proposition 17. *The NDOs* $(D, \{<\}, \{<\})$, $(D, \{<\}, \{\leq\})$, $(D, \{\leq\}, \{<\})$ *and* $(D, \{\leq\}, \{\leq\})$ *are safe.*

Proof. Like before, it is sufficient to provide a set $V \subseteq D$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally there exists no $r \in S^-$, which is satisfied by any $v \in V$. The following table gives such a $v \in V$ for every NDO (or a condition for it):

NDO	V
$(D, \{<\}, \{<\})$	$\max\{p \mid (<, p) \in S^-\} < v < \min\{p \mid (<, p) \in S^+\}$
$(D, \{<\}, \{\leq\})$	$v = \min\{p \mid (\leq, p) \in S^+\}$
$(D, \{\leq\}, \{<\})$	$\max\{p \mid (\leq, p) \in S^-\} < v < \min\{p \mid (<, p) \in S^+\}$
$(D, \{\leq\}, \{\leq\})$	$v = \min\{p \mid (\leq, p) \in S^-\}$

□

Proposition 18. *The NDOs* $(D, \{<\}, \{>\})$, $(D, \{\leq\}, \{>\})$, $(D, \{\leq\}, \{\geq\})$ *and* $(D, \{<\}, \{\geq\})$ *are safe.*

Proof. As previously, it is sufficient to provide a set $V \subseteq D$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally there exists no $r \in S^-$, which is satisfied by any $v \in V$. We supply a table that shows a $v \in V$ for every NDO (or a condition for it):

NDO	V
$(D, \{<\}, \{>\})$	$v > \max\{p \mid (<, p) \in S^- \ \vee \ (>, p) \in S^+\}$
$(D, \{\leq\}, \{>\})$	$v > \max\{p \mid (\leq, p) \in S^- \ \vee \ (>, p) \in S^+\}$
$(D, \{\leq\}, \{\geq\})$	$v > \max\{p \mid (\leq, p) \in S^- \ \vee \ (\geq, p) \in S^+\}$
$(D, \{<\}, \{\geq\})$	$v = \max\{p \mid (<, p) \in S^- \ \vee \ (\geq, p) \in S^+\}$

□

Proposition 19. *The NDOs* $(D, \{=\}, \{<\})$, $(D, \{=\}, \{\leq\})$, $(D, \{=\}, \{>\})$ *and* $(D, \{=\}, \{\geq\})$ *are safe.*

Proof. As done previously, it is sufficient to provide a set $V \subseteq D$, such that every $r \in S^+$ is satisfied by at least one $v \in V$ and, additionally there exists no $r \in S^-$, which is satisfied by any $v \in V$. The following table gives a $v \in V$ for every NDO by specifying the corresponding condition:

NDO	V
$(D, \{=\}, \{<\})$	$v < \min\{p \mid (<, p) \in S^+\} \ \wedge v \neq p$ for $(=, p) \in S^-$
$(D, \{=\}, \{\leq\})$	$v \leq \min\{p \mid (\leq, p) \in S^+\} \ \wedge v \neq p$ for $(=, p) \in S^-$
$(D, \{=\}, \{>\})$	$v > \max\{p \mid (>, p) \in S^+\} \ \wedge v \neq p$ for $(=, p) \in S^-$
$(D, \{=\}, \{\geq\})$	$v \geq \max\{p \mid (\geq, p) \in S^+\} \ \wedge v \neq p$ for $(=, p) \in S^-$

□

5.4 A tractability guide to NDOs

After dealing with concrete cases separately, we sum up the previous results in a single table. Tables 5.1 shows which NDOs are safe (noted with a ✓) and which are not (noted with an ✗). In these tables, D_1, D_2, D_3 are datatypes which have at least 1, 2 or 3 elements respectively, D_{nd} is a non-dense datatype and D_{2nd} is a double non-dense datatype. D_{nr} is a non-right-open datatype such that there exists a $q \in D_{nr}$ with $q < max_{D_{nr}}$ but there exists no $r \in D_{nr}$ with $q < r < max_{D_{nr}}$. Finally, D_d is a dense datatype and D_s is a sparse and right-open datatype. Both D_d and D_s are left- and right-open.

The symmetric NDOs of the ones that appear in Tables 5.1 are correspondingly safe or non-safe. E.g., $(D_2, \{\leq, \geq\}, \{\leq\})$ which is symmetric to $(D_2, \{\leq, \geq\}, \{\geq\})$ is non-safe as well. Additionally, if (D, O_-, O_+) is a safe

NDO	Safety	Proof page
$(D_2, \{\leq, \geq\}, \{\geq\})$	✗	52
$(D_2, \{\leq, \geq\}, \{>\})$	✗	52
$(D_2, \{\leq, >\}, \{\geq\})$	✗	52
$(D_2, \{\leq, >\}, \{>\})$	✗	52
$(D_3, \{<, >\}, \{\geq\})$	✗	52
$(D_3, \{<, >\}, \{>\})$	✗	52
$(D_1, \{<, =\}, \{\leq\})$	✗	52
$(D_{nd}, \{\leq, =\}, \{\leq\})$	✗	52
$(D_{nd}, \{<, =\}, \{<\})$	✗	52
$(D_{2nd}, \{\leq, =\}, \{<\})$	✗	52
$(D_{1r}, \{\leq, =\}, \{>\})$	✗	53
$(D_{1r}, \{<, =\}, \{>\})$	✗	53
$(D_{1r}, \{\leq, =\}, \{\geq\})$	✗	53
$(D_{1r}, \{<, =\}, \{\geq\})$	✗	53
$(D_d, \{\leq, =\}, \{<, >, \leq, \geq, =\})$	✓	53
$(D_d, \{<, \leq\}, \{<, >, \leq, \geq, =\})$	✓	54
$(D_d, \{<, \leq, =\}, \{<, >, \geq, =\})$	✓	55
$(D_s, \{<, \leq\}, \{<, >, \leq, \geq, =\})$	✓	56
$(D_s, \{<, \leq, =\}, \{>, \geq, =\})$	✓	57
$(D, \{<, >, \leq, \geq, =\}, \{=\})$	✓	58

Table 5.1: Maximal safe and minimal non-safe NDOs

NDO then (D', O'_-, O'_+) with $D' \subseteq D$ (with the properties of D being preserved in D'), $O'_- \subseteq O_-$ or $O'_+ \subseteq O_+$ is safe as well. In the same sense, if (D, O_-, O_+) is not a safe NDO then (D', O'_-, O'_+) with $D' \supseteq D$ (again, with the properties of D being preserved in D'), $O'_- \supseteq O_-$ or $O'_+ \supseteq O_+$ neither is it.

The safe NDOs we have presented are maximal, that is if one or more relational operator is added to O_- or O_+ they become non-safe. The maximality of the safe NDOs can be verified by the non-safe NDOs of table 5.1.

Similarly, the displayed non-safe NDOs are minimal, that is if one relational operator is removed from O_- or O_+ they become safe. The minimality of the non-safe NDOs can be verified using the propositions 17, 18 and 19. This is the reason why the NDOs of propositions 17, 18 and 19 are not included in the overall table; their safety has mainly been proved to support the minimality of the non-safe NDOs.

Finally, we remind that according to Proposition 7 the subsumption problem in $\mathcal{EL}^{\perp}(D, O_-, O_+)$, where (D, O_-, O_+) is a safe NDO is in PTIME. Theorem 22 on the other hand has proved that the subsumption problem in $\mathcal{EL}^{\perp}(D, O_-, O_+)$, with (D, O_-, O_+) being a non-convex NDO, is EXPTIME-hard. As a general conclusion, we observe that polynomiality is preserved

when relational operators that have the same direction are used on the left-hand side; however, polynomiality is lost when operators of opposite direction are used on the left-hand side with the case of $(D, \{<, >, \leq, \geq, =\}, \{=\})$ being an exception.

Table 5.1 along with the results of propositions 17, 18 and 19 do not provide a full classification of all the different NDOs that can appear. The number of all the possible NDOs is determined by the use of relational operators on the left- and right-hand side along with the different properties of the datatype (density, sparsity etc.). A thorough case analysis of this vast number is not a topic of the current work, but it might be a subject of future research.

Chapter 6

EL Profile in OWL 2

OWL 2 (Web Ontology Language) is a Semantic Web declarative language,[1] which is widely used for knowledge representation and reasoning purposes. It allows to model domain knowledge with adequate expressivity via the building of ontologies. Moreover, the availability of numerous implemented reasoning tools is a significant assistant factor for the modeling process. Additionally, the W3C standardization of OWL, which is the predecessor of OWL 2, assures the interoperability between the several ontologies expressed in OWL. OWL 2, which enriches OWL with additional features, is expected to be the new W3C standard version of OWL once it receives the final approval.

W3C has published a set of documents which strictly define the language (and an excellent guide to them)[2]. The documents specify the syntax and semantics of the language, as well as other relevant information such as useful sub-languages of OWL 2. OWL 2 EL Profile[3] is one of these sub-languages; it restricts the expressivity of OWL 2 for the sake of polynomial reasoning algorithms. The existing OWL 2 EL Profile corresponds to the most recent version of $\mathcal{EL}^{++}$ description logic [1, 2]. The main application area of OWL 2 EL Profile is considerably large ontologies for which basic expresivity is sufficient but polynomial reasoning is essential.

6.1 A Suggestion for a Datatype Extension

Our suggestion concerns the OWL 2 EL Profile; we attempt to introduce a small enhancement related to the datatypes part of the language. Thus, the work presented throughout this dissertation can be viewed as a practical contribution to this ontology language.

[1] `http://www.w3.org/TR/owl2-syntax/`
[2] `http://www.w3.org/TR/2009/WD-owl2-overview-20090611/#`
[3] `http://www.w3.org/TR/2009/CR-owl2-profiles-20090611/#OWL_2_EL`

OWL 2 [4] allows for extensive expressivity concerning the use of numerical datatypes. Specifically, it permits the use of multiple datatypes, such as reals or integers, in the same ontology. As a consequence, one can formulate concepts of the form $\exists \mathsf{hasAge}.18$ or $\exists \mathsf{hasTemperature}.37.5$. Apart from that OWL 2 offers support for limited datatypes, with the employment of *constraining facets*. Constraining facets restrict numerical datatypes to subsets of them by using relational operators such as $<$, $\leq$, $>$ and $\geq$. Consequently, concepts such as $\exists \mathsf{hasTemperature}.(>, 36.6)$ can be expressed.

However, the EL Profile of OWL 2, in an effort to retain polynomiality, reduces expressivity related to numerical datatypes. Similarly to OWL 2, multiple numerical datatypes, such as doubles or nationals, can be used; however, the corresponding constraining facets are now eliminated. Therefore, the ontology can contain concepts of the form $\exists \mathsf{hasAge}.18$ but not of the form $\exists \mathsf{hasAge}.(<, 18)$.

What we propose is a modification of the existing EL Profile in order to accommodate concepts of the form $\exists F.(op,q)$, where op is a relational operator (one of the $\{<, >, \leq, \geq, =\}$) and q is a data value. However, concepts of the form $\exists F.(op, q)$ cannot be used arbitrarily in either side of a subsumption axiom. There should be exact rules which specify which relational operators are used at which sides, obeying the restrictions that a safe NDO imposes. The proofs we have provided for the soundness, completeness and polynomiality of the consequence-driven reasoning algorithm guarantees that the so-desired tractability is not lost. The choice of the NDO may be different for each case, depending on the special needs of the aplication area.

However, the currently suggested framework still requires an extension in order to convey multiple NDOs, that is the use of different NDOs in the same ontology, provided that different NDOs are not shared by the same feature. The aforementioned extension is necessary in order to ensure compatibility with the current EL Profile and particularly its multiple datatypes. Since such an extension was not feasible in the current dissertation (mostly due to time and space limitations), it may well be part of future work.

If the EL Profile of OWL 2 is adapted in the abovementioned way, the constraining facets, which are available in OWL 2 at the moment, will be inserted in the EL Profile. As a consequence, numerical datatypes will be more precisely used in the modeling of knowledge. On the one hand ontology engineers will be able to formulate axioms of the form:

$$\mathsf{Child} \sqsubseteq \mathsf{hasAge}.(>, 5) \sqcap \mathsf{hasAge}.(<, 12)$$

and be more accurate in the representation of knowledge; on the other hand, given the axioms:

$$\mathsf{Child} \sqsubseteq \mathsf{hasAge}.(<, 12)$$

[4] `http://www.w3.org/TR/2009/CR-owl2-syntax-20090611/`

and

$$\mathsf{hasAge}.(<, 18) \sqsubseteq \mathsf{MinorDosageAllowed}$$

the axiom

$$\mathsf{Child} \sqsubseteq \mathsf{MinorDosageAllowed}$$

can be derived, improving, thus, the quality of the classification process. Hence, the above enrichment will enhance the restricted expressivity of OWL 2 EL Profile and enable the knowledge engineers to encode the domain knowledge in a more accurrate and meaningful way.

Chapter 7

Conclusions and Future Work

The present dissertation has produced results in the field of datatype DL reasoning and, particularly, for the numerical datatypes of the $\mathcal{EL}$ language. It has provided a set of inference rules for consequence-based reasoning in $\mathcal{EL}$ with numerical datatypes and formulated a sound and polynomial algorithm based on them. Furthermore, it has introduced and formally defined the notion of safety for numerical datatypes and proved completeness for the reasoning algorithm when the numerical datatype is safe. Additionally, it has attempted a classification of the several $\mathcal{EL}$ fragments with numerical datatypes in terms of safety and supplied a tractability guide to them. The dissertation as a whole can be seen as an effort to further sharpen the border between tractable and intractable fragments of $\mathcal{EL}$ w.r.t. numerical datatypes.

The current work presents several similarities with another work which involves the extension of $\mathcal{EL}$ with concrete domains (datatypes) [1]. In that approach, the property which datatypes should have in order to preserve polynomiality is p-admissibility. The p-admissibility property for datatypes assumes the satisfaction of two conditions: datatypes should be convex and satifiability and implication for the datatypes should be decided in polynomial time. If we compare with our approach we come up with two conclusions. Firstly, the safety condition which is introduced in this dissertation is a stricter and preciser version of convexity. Second, instead of assuming polynomiality for satifiability and implication for the datatypes we supply a full listing of exact cases where datatype expressions are unsatisfiable and where one datatype expression implies another. Additionally, we give specific instances of many polynomial $\mathcal{EL}$ fragments (not just examples) and, in general, provide a more fine-graned classification of tractable fragments concerning numerical datatypes.

Many difficulties were encountered throughout the completion of the

current project, the main of which was the various combinations of numerical datatypes with relational operators. If we take into consideration that five operators are examined, which can be used either on the left- or on the right-hand side of axioms, and the different properties that a datatype D can present, we end up with a great many possible fragments. Towards that end, we tried to detect maximal tractable fragments which can be used in practical applications and minimal intractable fragments which can be avoided. Furthermore, we explored the particular properties of datatypes that lead to (in)tractability. However, an exhaustive classification of the total amount of fragments neither was feasible nor among the main purposes of this dissertation.

Additionally, there are some issues that this dissertation did not deal with, primarily due to time restrictions. These matters might be questions of future research. For example, an assertional formalism should be developed for the above framework that will allow reasoning w.r.t. ABoxes. Apart from that, other useful datatypes may be considered which are used widely in practice such as strings. The notion of safety can be appropriately modified for strings or even generalised for all categories of datatypes. In addition to this, the current datatype approach could be applied in association with expressive description logics, such as $\mathcal{SHIQ}$ and the impact of this tractable extension of the language could be explored. Finally, the possibility of implementing the theoretical extension that this dissertation suggests and conducting experiments in order to compare with similar approaches might attract a research interest.

Bibliography

[1] 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, 2005. Morgan-Kaufmann Publishers.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jõrg Siekmann, Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2003.

[5] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[6] Ronald J. Brachman, Alexander Borgida, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lori Alperin Resnick. The classic knowledge representation system or, kl-one: The next generation. In *Preprints of the Workshop on Formal Aspects of Semantic Networks, Two Harbors*, pages 1036–1043. MorganKaufman, 1989.

[7] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. López de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press, 2004.

[8] C.Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, 2002.

[9] Yevgeny Kazakov. Consequence-driven reasoning for horn $\mathcal{SHIQ}$ ontologies. In *Proc. of IJCAI 2009*, 2009. In press.

[10] Berners T. Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[11] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning computational intelligence journal. 3:78–93, 1987.

[12] Carsten Lutz. Description logics with concrete domains - a survey, 2003.

[13] Boris Motik and Ian Horrocks. Owl datatypes: Design and implementation. In *7th International Semantic Web Conference (ISWC2008)*, October 2008.

[14] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[15] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning*, pages 371–443. 2001.