Charles University, Prague
Faculty of Mathematics and Physics

MASTER THESIS

Modelování n-árních relací v deskripčních logikách

Modelling n-ary relations in description logics

Miroslav Blaško

Department of Software Engineering
Supervisor: RNDr. Filip Zavoral, Ph.D.
Study Program: Computer Science

Prague 2008

Acknowledgements

I would like to thank my supervisor RNDr. Filip Zavoral, Ph.D. for his overall support. My big thanks belongs to my friends for their priceless remarks.

Abstrakt

Názov práce: Modelování n-árních relací v deskripčních logikách
Autor: Miroslav Blaško
Katedra: Katedra softwarového inženýrstvii
Vedúci diplomovej práce: RNDr. Filip Zavoral, Ph.D.
E-mail vedúceho: filip.zavoral@mff.cuni.cz
Abstrakt:

$\mathcal{DLR}$ je expresívna deskripčná logika, ktorá podporuje n-árne relácie. V súčastnosti neexistuje algoritmus, ktorý by dokázal natívne uvažovať v $\mathcal{DLR}$. Existujú však dve práce, ktoré umožňujú uvažovanie delegovať do binárnych logík.

V tejto práci definujeme novú deskripčnú logiku $\mathcal{NDL}$. Tá predstavuje podmnožinu $\mathcal{DLR}$, pre ktorú veríme, že natívne uvažovanie vieme poskytnúť. Na základe spomínaných prací vytvoríme transformácie z $\mathcal{NDL}$ do binárnych logík, ktoré budeme mocť použiť v najmodernejších odvodzovacích systémoch. Nové transformácie teoreticky i prakticky analyzujeme. N-árne data pre testovanie vytvoríme z existujúcich OWL ontológií opačnou transformáciou.

Táto práca môže byť použita pre porovnanie natívneho uvažovania a uvažovania pomocou transformácie do binárnych logík.

Kľúčové slová: *n-árne deskripčné logiky, reifikácia*

Abstract

Title: Modelling n-ary relations in description logics
Author: Miroslav Blaško
Department: Department of Software Engineering
Supervisor: RNDr. Filip Zavoral, Ph.D.
Supervisor's e-mail address: filip.zavoral@mff.cuni.cz
Abstract:

$\mathcal{DLR}$ is an expressive description logic with support of n-ary relations. Currently, there is no known algorithm for native reasoning within $\mathcal{DLR}$. However there are two approaches that allow to delegate reasoning services of $\mathcal{DLR}$ to binary description logics.

In this work we define new description logic $\mathcal{NDL}$, a subset of $\mathcal{DLR}$, for which we believe that native reasoning can be provided. Based on the existing approaches, we transform $\mathcal{NDL}$ to binary description logics for which the current of state-of-art of reasoners exist. New transformations will be analysed both theoretically and empirically. N-ary data for benchmark will be created from existing OWL ontologies by transformation of opposite direction.

This benchmark can be used for comparison of native reasoning and reasoning by transformation to binary DLs.

Keywords: *n-ary description logics, reification*

Contents

1 Introduction and Overview

1.1 Semantic Web and Annotation Creation

Semantic Web is an extension of the current World Wide Web, in which meaning of data is captured in machine-readable form. This formal knowledge is captured by semantic metadata, thus allowing easier exchange, integration and processing of data. Usually, data description is separated from original data, making it possible to have different views on data for different applications.

Semantic annotation authoring is a process of creating metadata upon existing documents. Since vitality of semantic web is highly dependent on both quality and mass of metadata, annotation creation became one of the basic milestones of semantic web realization. Although there are many techniques that allow fully automated annotation creation from data, most applications need some knowledge which is created manually. For authoring of sufficient amount of metadata, tools for annotation creation must be easy to learn, not only for knowledge engineers, but also for domain experts. Ontology language, i.e. formal specification of annotations, must be simple, straightforward and expressive enough. Furthermore, to ensure good quality of metadata, the tools have to support detection and debugging of logical modelling errors.

One of such annotation tools is a document annotation tool DNAT (Dynamic Narrative Authoring Tool) developed within European project CIPHER (Communities of Interest Promoting Heritage of European Regions) [8]. DNAT was designed to support users in creating printable knowledge-intensive content and the corresponding knowledge-base at the same time. DNAT enables users to link created annotations to text fragments in the narrative using semantic annotations created via Conceptual Graphs (CG) [12]. Then, CG represents easily readable model of the narrative and shows the dependencies of narrative primitives.

Experiments proved that CGs in DNAT were very well suited for document annotation. Authors of DNAT discovered that most of the people learned to model knowledge in a few hours, since annotating process was very similar to using color markers for highlighting key concepts in documents. Unfortunately for DNAT to provide ontology services and query support, also complex inference support over gathered data was required. However, general CGs are equivalent to FOL (first order logic) and therefore deduction in CGs is not decidable.

During the knowledge extraction process, DNAT automatically builds a knowledge base in Apollo CH format [22]. The format allows to export knowledge base to other formalisms, for which suitable reasoning services might exists. From all supported formalisms, OWL, in particular OWL-DL was most likely the best candidate to fulfill requirements of DNAT.

OWL (Web Ontology Language) is an expressive ontology language released as a W3C (World Wide Web Consortium) recommendation in February 2004. OWL-DL is a subset of OWL that was designed to support highest possible expressivity while retaining decidability. The name OWL-DL is due to its correspondence to description logics (DL) [23, 25]. In DLs, each standard reasoning task can be reduced to satisfiability problem of a DL knowledge base, i.e. checking if there is a model of the knowledge base in which all axioms evaluate to true.

Further investigation on annotating with OWL-DL revealed a significant weakness of OWL. Even though OWL-DL is very expressive, it does not have direct support for n-ary relation constructors. In July 2004, W3C published first informative document about defining n-ary relations [6] in OWL. Document explains how to model n-ary relations by so-called reification, i.e. representing n-ary relation by concept. Such a representation is, however, weak form of reification in which n-ary relation does not have to be interpreted as a set of tuples.

As natural choice the DNAT seems to profit from an n-ary extension like the logic formalism $\mathcal{DLR}$ [35]. $\mathcal{DLR}$ is description logic with support of n-ary relations. There is no known algorithm for native reasoning within $\mathcal{DLR}$. However, in the work of Calvanese at al. [38], $\mathcal{DLR}$ was used for solving query containment problem (QCP). Part of the solution of QCP was satisfiability problem of $\mathcal{DLR}$ which was translated and delegated to satisfiability problem of $\mathcal{CIQ}$ DL knowledge base. Since there was no implementation of reasoner based on $\mathcal{CIQ}$ DL, Calvanese at al. solution did not lead to practical decidability. To overcome this problem, Horrocks et al. introduced slightly different mapping of QCP to $\mathcal{DLR}$ and translated it to satisfiability problem of a $\mathcal{SHIQ}$ DL knowledge base. In both algorithms reification was used for transformation of $\mathcal{DLR}$ to binary DLs, however, some additional axioms were added to the knowledge bases to axiomatize all properties of n-ary relations correctly.

1.2 Goal of the Thesis

Currently, authors of DNAT restricted their attention to a subset of $\mathcal{DLR}$ that seems suitable for native tableaux reasoning [24]. It allows for all $\mathcal{DLR}$ constructs accept number restrictions that represent a significant source of non-determinism in tableaux reasoning. On this subset of $\mathcal{DLR}$ we defined new description logic $\mathcal{NDL}$. In addition, we extended $\mathcal{NDL}$ with new constructors to fulfill semantic annotation needs. New constructors, however, brings only syntactic sugar to the language.

The main goal of this thesis is to design and implement reification algorithms for the logic $\mathcal{NDL}$ based on approaches of Calvanese at al. and Horrocks at al. Then, the algorithms will be compared both theoretically and empirically. Empirical testing will be done on the current state-of-art of reasoners with appropriate n-ary $\mathcal{NDL}$ knowledge bases. $\mathcal{NDL}$ knowledge bases will be gathered from existing OWL ontologies.

This benchmark can be used for comparison of native reasoning and reasoning by transformation to binary DLs.

1.3 Structure of the Text

The theoretical background and overview of $\mathcal{DLR}$, OWL-DL and its underlying logics is described in Chapter 2. Overview of state-of-art ontology languages and its issues for semantic annotation in project DNAT is described in Chapter 3. Syntax and semantics of new n-ary description logics $\mathcal{NDL}$ is introduced in Chapter 4 along with its graphical notation and example. Reification algorithms for $\mathcal{NDL}$ are described in Chapter 5. Transformations of OWL into $\mathcal{NDL}$ is described in Chapter 6. Theoretical analysis of complexity of transformed $\mathcal{NDL}$ KBs and its empirical results on the current state-of-art of reasoners are explained in Chapter 7. Contribution of this work and open issues are summarized in Chapter 8.

2 Foundations

Technical background around semantic web is new and in some cases misunderstood. This chapter is dedicated to clarify ambiguous terminology, describe syntax and semantics of languages that will be used in further text. In the first section of the chapter, difference between ontology and knowledge base is explained. In Section 2.2 a family of logic-based knowledge representation formalisms, Description Logics is introduced. Next two sections explains concrete types of Descriptions Logics : first, underlying logics of OWL-DL is in Section 2.3; second, the logics $\mathcal{DLR}$ in Section 2.4. Last section of the chapter describes abstract syntax of OWL and explains its correspondence to description logics.

2.1 Ontologies and Knowledge Bases

Word "ontology" generates a lot of controversial discussion about AI, because its meaning is a bit vague and as the term it is used in many different ways. For the AI community, definition proposed in 1992 by Tom Gruber describes ontology as "a specification of conceptualization". Thus, an ontology is a description (like a formal specification of a program) of the concepts and relations that can exist. However, in other resources, to the term it might be referred as a philosophical discipline, an informal conceptual system, a formal semantic account, a representation of a conceptual system via a logical theory, the vocabulary used by a logical theory as well as (meta-level) specification of a logical theory [9].

Word "knowledge base" is more general term. In context of semantic web, knowledge base usually refers to an informal term for a collection of information that includes an ontology as one component. Besides an ontology, a knowledge base may contain information specified in a declarative language such as logic or expert-system rules, but it may also include unstructured or unformalized information expressed in natural language or procedural code.

Within knowledge engineering community, the term "ontology" has recently gained popularity, most likely because of raising popularity of OWL, that defines the term in the specification of language. In further text we will use term "ontology", only in context of OWL, as a set of OWL axioms. On the other hand, whenever word "knowledge base" is used, it will refer to collection of axioms in description logics (see Section 2.2). Hence, when describing OWL, one can refer to its ontology or to knowledge base of its underlying description logics.

2.2 Description logics

Description logics (DLs) [23], [25] are a family of logic-based knowledge representation formalisms. They are usually a (decidable) subset of First Order Predicate Logic (FOL), and thus have a well-defined, formal semantics.

The basic building blocks of DLs are atomic concepts, typically atomic binary relations, called roles and individuals. Atomic concepts and roles correspond to unary and binary predicates in FOL. Concepts denote set or a class of objects and are mainly used to define a domain of application. Relations between objects is described via roles. Complex concepts and relations can be formed from atomic ones, using DL operators called constructors. Individuals correspond to constants in FOL and they represent objects in the domain.

DL knowledge base is set of logical sentences, called axioms. It can be divided into three components: *TBox*, *RBox*, and *ABox*. The *TBox* contains statements about concepts (e.g. concept subsumption $Man \sqsubseteq Person$). The *RBox* contains statements about roles (e.g. *hasBrother* is symmetric role) and role hierarchies (e.g. role subsumption $hasSon \sqsubseteq hasChild$). The *ABox* defines role assertions between individuals (e.g. $hasChild(Peter, John)$) and concept membership assertions (e.g. $John : Man$). In some literature, component *RBox* is omitted and statements about roles are included in set of terminological axioms, i.e. *TBox*. This kind of partitioning of KB will be used in the rest of this text.

One of most important categorizing criteria of description logics is its expressivity, i.e. set of constructors and axiom types that are allowed to occur in knowledge base. To encode the precise expressivity of the particular description logic, mnemonic names were assigned. For a list of mnemonics with DL's they characterize, see Table 2.1.

In the table, for example mnemonics $\mathcal{AL}$ stands for the most simple DL called Attribute Logics. In logic $\mathcal{AL}$, atomic concepts and atomic roles can be used, and among all concept constructors only following are permitted: negation of atomic concept ($\neg A$), intersection of two concepts ($C \sqcap D$) and other two constructors ($\exists \mathbf{r}.\top$) and ($\forall \mathbf{r}.C$). All of the constructors in the

Mnemonic	**DL Expressivity**
$\mathcal{AL}$	Attribute Logic [A, $\neg A$ (atomic), $C \sqcap D$, $\exists\mathbf{r}.\top$, $\forall\mathbf{r}.C$]
$\mathcal{ALC}$	Attribute Logic + Full Complement [allowing $C \sqcup D$ and $\exists\mathbf{r}.C$]
$\mathcal{R}+$	Transitive Roles
$\mathcal{S}$	$\mathcal{ALCR}+$
$\mathcal{H}$	Role Hierarchy
$\mathcal{I}$	Inverse Roles
$\mathcal{O}$	Nominals (individuals used in class expressions)
$\mathcal{N}$	Unqualified Cardinality Restriction [$\leqslant n\mathbf{r}$, $\geqslant n\mathbf{r}$, $= n\mathbf{r}$]
$\mathcal{Q}$	Qualified Cardinality Restriction [$(\leqslant n\mathbf{r}).C$, $(\geqslant n\mathbf{r}).C$, $(= n\mathbf{r}).C$]
$\mathcal{D}$	Datatypes
$\mathcal{F}$	Functional Roles

Table 2.1: Mnemonics for DLs

table will be explained later in the text.

With restriction that $\mathcal{AL}$ DL has to be included in every DL, mnemonics from Table 2.1 can be combined to form more complex DLs. For example description logics $\mathcal{ALIF}$ would denote Attribute Logics extended with inverse role constructors ($\mathcal{I}$) and axioms that define functional role ($\mathcal{F}$).

Following two sections will introduce two concrete logics from the family of description logics that are used in the rest of this text. First, the logic $\mathcal{SHOIQ}$ DL that is underlying logic of OWL-DL extended with qualified number restrictions (mnemonic $\mathcal{Q}$). This extension of OWL-DL will be used in next chapters as most of the reasoners already support it. Second, the logic $\mathcal{DLR}$ that is not a standard DL and therefore mnemonics from the Table 2.1 cannot be applied to it.

2.3 The Logic SHOIQ

$\mathcal{SHOIQ}$ [28] is a standard DL, in the sense that it deals with concepts and (only) binary relations (called roles). However it is very expressive as it supports individuals in concept constructors, inverse roles, qualified number restrictions on roles, transitive roles, role inclusion axioms and so on. In following text syntax and semantics of $\mathcal{SHOIQ}$ will be described. Some of definitions were taken from [44].

2.3.1 Syntax

Definition 2.3.1 ($\mathcal{SHOIQ}$ concepts, relations and individuals). *Given a set of atomic concept names* NC*, a set of atomic role names* NR *with transitive role names* $\mathsf{NR}_+ \subseteq \mathsf{NR}$*, and a set of individuals* NI*, every* $C \in \mathsf{NC}$ *is a concept, every* $\mathbf{r} \in \mathsf{NR}$ *is a role, and every* $\mathbf{r} \in \mathsf{NR}_+$ *is a transitive role. If* $\mathbf{r}$ *is a role, then* $\mathbf{r}^-$ *is also a role (and if* $\mathbf{r} \in \mathsf{NR}_+$ *then* $\mathbf{r}^-$ *is also a transitive role). If* $\mathbf{s}$ *is a (possibly inverse) role,* C*,* D *are concepts,* o *is an individual, and* k *is a non-negative integer, then*

$$\begin{array}{c} \bot, \top, \neg C, C \sqcap D, C \sqcup D, \\ \exists \mathbf{s}.C, \forall \mathbf{s}.C, \leq k\mathbf{s}.C, \geq k\mathbf{s}.C, = k\mathbf{s}, \{o\} \end{array} \quad \textit{are } \mathcal{SHOIQ} \textit{ concepts.}$$

A $\mathcal{SHOIQ}$ knowledge base consist of schema and *ABox*. Schema is a set of logical implication axioms, which is used to express general facts about knowledge base. *ABox* is a set of axioms asserting facts about individuals and pairs of individuals.

Definition 2.3.2 ($\mathcal{SHOIQ}$ schema). *A* $\mathcal{SHOIQ}$ *schema* $\mathcal{S}$ *is a set of axioms of the form* $C \sqsubseteq D$ *and* $\mathbf{r} \sqsubseteq \mathbf{s}$*, where* C*,* D *are* $\mathcal{SHOIQ}$ *concepts and* $\mathbf{r}$*,* $\mathbf{s}$ *are* $\mathcal{SHOIQ}$ *roles.*

Definition 2.3.3 ($\mathcal{SHOIQ}$ *ABox*). *A* $\mathcal{SHOIQ}$ *ABox* $\mathcal{A}$ *is a set of axioms of the form* $w : C$ *and* $\langle v, w \rangle : \mathbf{r}$*, where* C *is a concept,* $\mathbf{r}$ *is a role, and* v*,* w *are individuals.*

Definition 2.3.4 ($\mathcal{SHOIQ}$ knowledge base). *A* $\mathcal{SHOIQ}$ *knowledge base* $\mathcal{K}$ *is a pair* $\langle \mathcal{S}, \mathcal{A} \rangle$*, where* $\mathcal{S}$ *is a schema and* $\mathcal{A}$ *is an ABox.*

2.3.2 Semantics

Semantics of $\mathcal{SHOIQ}$ is given by interpretation.

Definition 2.3.5 ($\mathcal{SHOIQ}$ interpretation). *Given a* $\mathcal{SHOIQ}$ *knowledge base* $\mathcal{K}$*, an interpretation is a pair* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *, where* $\Delta^{\mathcal{I}}$ *is the domain (a non-empty set), and* $\cdot^{\mathcal{I}}$ *is an interpretation function that maps every concept to a subset of* $\Delta^{\mathbf{I}}$*, every role to a subset of* $(\Delta^{\mathcal{I}})^2$*, and every individual to an element in* $\Delta^{\mathbf{I}}$ *such that the following equations are satisfied.*

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \{\} \\
C^{\mathcal{I}} &\subseteq \top^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\{o\})^{\mathcal{I}} &= \{o^{\mathcal{I}}\} \\
o^{\mathcal{I}} &\in \top^{\mathcal{I}}
\end{aligned}
\qquad
\begin{aligned}
\mathbf{r}^{\mathcal{I}} &= (R^{\mathcal{I}})^{+} \textit{ for all } \mathbf{r} \in \mathsf{NR}_{+} \\
(\mathbf{r}^{-})^{\mathcal{I}} &= \{(d, d') \mid (d, d') \in \mathbf{r}^{\mathcal{I}}\} \\
(\exists \mathbf{s}.C)^{\mathcal{I}} &= \{d \mid \exists (d, d') \in \mathbf{s}^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\} \\
(\forall \mathbf{s}.C)^{\mathcal{I}} &= \{d \mid \forall (d, d') \in \mathbf{s}^{\mathcal{I}} \Rightarrow d' \in C^{\mathcal{I}}\} \\
(\leq k\mathbf{s}.C)^{\mathcal{I}} &= \{d \mid \sharp\{\exists d'.(d, d') \in \mathbf{s}^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\} \leq k\} \\
(\geq k\mathbf{s}.C)^{\mathcal{I}} &= \{d \mid \sharp\{\exists d'.(d, d') \in \mathbf{s}^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\} \geq k\} \\
(= k\mathbf{s}.C)^{\mathcal{I}} &= \{d \mid \sharp\{\exists d'.(d, d') \in \mathbf{s}^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\} = k\}
\end{aligned}$$

where C, D are concepts, $\mathbf{r}$, $\mathbf{s}$ *are roles, o is individual, and k is a non-negative integer.*

In every DL knowledge base there are at least two concepts : *the most general concept* ($\top$) and *the most specific concept* $\bot$. For every concept C in KB it is true that $C \sqsubseteq \top$ and $\bot \sqsubseteq C$.

$\mathcal{SHOIQ}$ DL knowledge base contains 3 logical concept constructors: *negation of concept* ($\neg C$), *intersection of concepts* ($C \sqcap D$) and *union of concepts* ($C \sqcup D$). From all the constructors, only *nominals* ($\{o\}$) can use individuals in the concept constructor. Thanks to *nominals*, concepts of $\mathcal{SHOIQ}$ DL can be forced to contain not empty, but finite set of individuals (e.g. KB containing only two axioms: $A \sqsubseteq \{o\}, \{o\} \sqsubseteq A$, forces concept A to have exactly one individual o). Note that without *nominals*, it is not possible.

To all other concept constructors it is referred to as *role restrictions. An existential restriction* $\exists \mathbf{s}.C$ describes a concept of individuals that have at least one relationship along role $\mathbf{s}$ to an individual that is a member of concept C. *An universal restriction* $\forall \mathbf{s}.C$ describes a concept of individuals that have relationships along role $\mathbf{s}$ to only individuals that are members of concept C. *Existential restrictions* and *universal restrictions* are sometimes also called *qualifier restrictions.*

With *cardinality restrictions* it is possible to describe concept of individuals that have at least, at most or exactly a specified number of relationships with other individuals of given concept. *A minimum* (*maximum*, *exact*) *cardinality restriction* $\geq k\mathbf{s}.C$ ($\leq k\mathbf{s}.C$, $= k\mathbf{s}.C$) specifies that an individual must participate in at least (at most, exactly) k of $\mathbf{s}$ relationships to members of concept C. If the concept C is the most general concept, i.e. $\top$, cardinality restrictions are called *unqualified.* In all other cases they are *qualified cardinality restrictions.*

Definition 2.3.6 (Satisfiability of knowledge base, schema, *ABox*, axioms). *Given a $\mathcal{SHOIQ}$ knowledge base $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$, with concepts C, D, roles $\mathbf{r}$, $\mathbf{s}$, individuals v, w, an interpretation $\mathcal{I}$ satisfies axioms according to following conditions. $\mathcal{I}$ satisfy $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; $\mathcal{I}$ satisfy $\mathbf{r} \sqsubseteq \mathbf{s}$ iff $\mathbf{r}^{\mathcal{I}} \subseteq \mathbf{s}^{\mathcal{I}}$; $\mathcal{I}$ satisfy $w : C$ iff $w^{\mathcal{I}} \in C^{\mathcal{I}}$; $\mathcal{I}$ satisfy $\langle v, w \rangle : \mathbf{r}$ iff $(v^{\mathcal{I}}, w^{\mathcal{I}}) \in \mathbf{r}^{\mathcal{I}}$.*

Additionally, the interpretation $\mathcal{I}$ satisfies a schema $\mathcal{S}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{S}$, satisfies an ABox $\mathcal{A}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{A}$, and satisfies a KB $\mathcal{K}$ iff it satisfies both $\mathcal{S}$ and $\mathcal{A}$.

If the interpretation $\mathcal{I}$ satisfies a concept, axiom, schema, or ABox X, then we say that $\mathcal{I}$ is a model of X, call X satisfiable, and write $\mathcal{I} \models X$.

Note that it is not assumed that individuals with different names are mapped to different elements in the domain (the so-called unique name assumption).

2.4 The Logic DLR

$\mathcal{DLR}$ is one of few description logics with ability to describe relations of any arity. It was first introduced in [45]. Distinguishing feature of $\mathcal{DLR}$ from any other DL formalism is capability of expressing inclusion axioms on complex relations. In following text syntax and semantics of $\mathcal{DLR}$ will be described. Most of the definitions here can be found in [44].

2.4.1 Syntax

Definition 2.4.1 ($\mathcal{DLR}$ concepts, relations and individuals). *Given a set of atomic concept names NC, a set of atomic relation names NR, and a set of individuals NI, every $C \in \mathsf{NC}$ is a concept, every $\mathbf{R} \in \mathsf{NR}$ is a relation, with every $\mathbf{R}$ having an associated arity, and every $w \in \mathsf{NI}$ is an individual. If C, D are concepts, $\mathbf{R}$, $\mathbf{S}$ are relations of arity n, i is an integer $1 \leq i \leq n$ then*

$$\begin{array}{rl} \top,\ \neg C,\ C_1 \sqcap C_2,\ \leq k[\$i]\mathbf{R},\ \forall[\$i]\mathbf{R} & \textit{are } \mathcal{DLR} \textit{ concepts, and} \\ \top_n,\ \neg\mathbf{R},\ \mathbf{R_1} \sqcap \mathbf{R_2},\ (\$i/n : C) & \textit{are } \mathcal{DLR} \textit{ relations with arity } n \end{array}$$

Relation expressions must be well-typed in the sense that only relations with the same arity can be conjoined, and in constructs like $\leq k[\$i]\mathbf{R}$ and $\forall[\$i]\mathbf{R}$ the value of i must be less than or equal to the arity of $\mathbf{R}$ and the value of k must be greater than or equal to zero.

A $\mathcal{DLR}$ knowledge base consist of schema and *ABox*. Schema is set of logical implication axioms, that is used to express general facts about knowledge base. *ABox* is a set of axioms asserting facts about individuals and tuples of individuals.

Definition 2.4.2 ($\mathcal{DLR}$ schema)**.** *A $\mathcal{DLR}$ schema $\mathcal{S}$ is a set of axioms of the form $C \sqsubseteq D$ and $\mathbf{R} \sqsubseteq \mathbf{S}$, where C, D are $\mathcal{DLR}$ concepts and $\mathbf{R}$, $\mathbf{S}$ are $\mathcal{DLR}$ relations of the same arity.*

Definition 2.4.3 ($\mathcal{DLR}$ *ABox*)**.** *A $\mathcal{DLR}$ ABox $\mathcal{A}$ is a set of axioms of the form $w : C$ and $\boldsymbol{w} : \mathbf{R}$, where C is a concept, $\mathbf{R}$ is a relation of arity n, w is an individual and $\boldsymbol{w}$ is an n-tuple $\langle w_1, ..., w_n\rangle$ such that $w_1, ..., w_n$ are individuals.*

Definition 2.4.4 ($\mathcal{DLR}$ knowledge base)**.** *A $\mathcal{DLR}$ knowledge base $\mathcal{K}$ is a pair $\langle \mathcal{S}, \mathcal{A}\rangle$, where $\mathcal{S}$ is a schema and $\mathcal{A}$ is an ABox.*

2.4.2 Semantics

Semantics of $\mathcal{DLR}$ is given by interpretation.

Definition 2.4.5 ($\mathcal{DLR}$ Interpretation)**.** *Given a $\mathcal{DLR}$ knowledge base $\mathcal{K}$, an interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain (a non-empty set), and $\cdot^{\mathcal{I}}$ is an interpretation function that maps every concept to a subset of $\Delta^{\mathbf{I}}$, every n-ary relation to a subset of $(\Delta^{\mathcal{I}})^n$, and every individual to an element in $\Delta^{\mathbf{I}}$ such that the following equations are satisfied.*

$$\begin{array}{rlrl}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} & \top_n^{\mathcal{I}} &\subseteq (\Delta^{\mathcal{I}})^n \\
C^{\mathcal{I}} &\subseteq \top^{\mathcal{I}} & \mathbf{R}^{\mathcal{I}} &\subseteq \top_n^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} & (\neg \mathbf{R})^{\mathcal{I}} &= \top_n^{\mathcal{I}} \backslash \mathbf{R}^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} & (\mathbf{R} \sqcap \mathbf{S})^{\mathcal{I}} &= \mathbf{R}^{\mathcal{I}} \cap \mathbf{S}^{\mathcal{I}} \\
w^{\mathcal{I}} &\in \top^{\mathcal{I}} & \boldsymbol{w}^{\mathcal{I}} &\in \top_n^{\mathcal{I}}
\end{array}$$

$$\begin{aligned}
(\forall[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \forall (d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}} \Rightarrow d_i = d\} \\
(\leq k[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \#\{(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}} \mid d_i = d\} \leq k\} \\
(\$i/n : C)^{\mathcal{I}} &= \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}
\end{aligned}$$

where C, D are concepts, $\mathbf{R}$, $\mathbf{S}$ are relations of arity n, w is individual, $\boldsymbol{w}$ is n-tuple of individuals, and i is an integer $1 \leq i \leq n$.

Note that $\top_n$ does not need to be interpreted as the set of all tuples of arity n, but only as a subset of them, and that the negation of a relation $\mathbf{R}$ with arity n is relative to $\top_n$.

Similarly as in binary DLs, $\mathcal{DLR}$ contains the most general concept $\top$. However $\mathcal{DLR}$ also contains $\top_n$ that represent the most general relation of arity n. Thus, in every KB, for each relation $\mathbf{R}$ of arity n occurring in KB it is true that $\mathbf{R} \sqsubseteq \top_n$.

From the complex concept constructors, $\mathcal{DLR}$ contains *a negation of concept* $(\neg C)$, *an intersection of concepts* $(C_1 \sqcap C_2)$, *an universal restriction* $(\forall[\$i]\mathbf{R})$ and *a maximal cardinality restriction* $(\leq k[\$i]\mathbf{R})$.

From the complex relation constructors, $\mathcal{DLR}$ contains *a negation of relation* $(\neg\mathbf{R})$, *an intersection of relations* $(\mathbf{R_1} \sqcap \mathbf{R_2})$, *an projection* $(\$i/n : C)$. Projection $\$i/n : C$ describes set of all tuples from $\top_n$, from which i-th individual is member of concept C.

Definition 2.4.6 (Satisfiability of knowledge base, schema, *ABox*, axioms)**.** *Given a $\mathcal{DLR}$ knowledge base $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$, with concepts C, D, relations $\mathbf{R}$, $\mathbf{S}$ of arity n, individual w and n-tuple of individuals $\boldsymbol{w}$, an interpretation $\mathcal{I}$ satisfies axioms according to following conditions. $\mathcal{I}$ satisfy $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; $\mathcal{I}$ satisfy $\mathbf{R} \sqsubseteq \mathbf{S}$ iff $\mathbf{R}^{\mathcal{I}} \subseteq \mathbf{S}^{\mathcal{I}}$; $\mathcal{I}$ satisfy $w : C$ iff $w^{\mathcal{I}} \in C^{\mathcal{I}}$; $\mathcal{I}$ satisfy $\boldsymbol{w} : \mathbf{R}$ iff $\boldsymbol{w}^{\mathcal{I}} \in \mathbf{R}^{\mathcal{I}}$.*

Additionally, the interpretation $\mathcal{I}$ satisfies a schema $\mathcal{S}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{S}$, satisfies an ABox $\mathcal{A}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{A}$, and satisfies a KB $\mathcal{K}$ iff it satisfies both $\mathcal{S}$ and $\mathcal{A}$.

If the interpretation $\mathcal{I}$ satisfies a concept, axiom, schema, or ABox X, then we say that $\mathcal{I}$ is a model of X, call X satisfiable, and write $\mathcal{I} \models X$.

Note that it is not assumed that individuals with different names are mapped to different elements in the domain.

2.5 OWL-DL

Based on family of description logics, ontology language OWL-DL was designed to support highest possible expressivity while retaining decidable. More precisely, OWL-DL is based on the $\mathcal{SHOIQ}$ DL [29] with restricted form of number restrictions to be unqualified (see [24]). It adds a simple form of Datatypes (often called concrete domains in DLs [30]). Following the usual DL mnemonics shown in Table 2.1, the resulting logic is called $\mathcal{SHOIN}(\mathcal{D})$.

Abstract syntax of OWL-DL [7] is very similar to description logics syntaxes. Description logics knowledge base is analogous to ontology in OWL-DL, DL concepts corresponds to classes, roles corresponds to individualvalued properties. Individuals are same in both terminologies, but sometimes OWL-DL refers to them as instances as well.

In addition, OWL-DL contains data ranges, which are analogous to concrete datatypes in description logics. Properties of OWL are of two types. Datatype property associates an individual with data value. Object property, also called abstract property, associates pairs of individuals. Domain of classes and data ranges is disjoint, also properties and constructors work on classes and data ranges separately.

Considering data ranges and datatype properties in this thesis would not lead to any challenging complications, as most of the time, data ranges can be treated in similar way as classes, and datatype properties as object properties. However it would lead to more confusing definitions of algorithms, theorems and so on. For this reason, existence of data values, data ranges and datatype properties in this thesis is omitted and it is referred to OWL-DL as subset of $\mathcal{SHOIN}$ DL.

OWL abstract syntax	DL syntax
Class(A)	A
Class(owl:Thing)	$\top$
Class(owl:Nothing)	$\bot$
complementOf(C)	$\neg C$
intersectionOf(C_1 ... C_n)	$C_1 \sqcap ... \sqcap C_n$
unionOf(C_1 ... C_n)	$C_1 \sqcup ... \sqcup C_n$
oneOf(o_1 ... o_n)	$\{o_1, ..., o_n\}$
restriction(**r** allValuesFrom(C))	$\forall \mathbf{r}.C$
restriction(**r** someValuesFrom(C))	$\exists \mathbf{r}.C$
restriction(**r** value(o))	$\exists \mathbf{r}.\{o\}$
restriction(**r** minCardinality(n))	$\geqslant n\mathbf{r}$
restriction(**r** maxCardinality(n))	$\leqslant n\mathbf{r}$
restriction(**r** cardinality(n))	$\geqslant n\mathbf{r} \sqcap \leqslant n\mathbf{r}$

Table 2.2: OWL-DL classes in DL abstract syntax

Complex classes, also called class descriptions, are formed in OWL-DL using class constructors. Simplified overview (without loss of expressivity) of all class constructors and its corresponding constructs in DL syntax are provided in Table 2.2; in the table A is a class name (atomic concept), $owl : Thing$ is superclass of all classes, $owl : Nothing$ is most specific class, C (possibly subscripted) is a class, **r** is object property or possibly inverse of object property, o (possibly subscripted) is an individual, n is non-negative integer. For more precise description of class constructors see OWL documentation [4, 7].

OWL abstract syntax	DL syntax
Class(A partial C_1 ... C_n)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n$
Class(A complete C_1 ... C_n)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n, C_1 \sqcap ... \sqcap C_n \sqsubseteq A$
EnumeratedClass(A o_1 ... o_n)	$A \sqsubseteq \{o_1, ..., o_n\}, \{o_1, ..., o_n\} \sqsubseteq A$
DisjointClasses(C_1 ...C_n)	$C_i \sqsubseteq \neg C_j, 1 \leq i < j \leq n$
EquivalentClasses(C_1 ... C_n)	$C_i = C_{i+1}, 1 \leq i < n$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
ObjectProperty($\mathbf{r}$ super($\mathbf{r_1}$))	$\mathbf{r} \sqsubseteq \mathbf{r_1}$
ObjectProperty($\mathbf{r}$ inverseOf($\mathbf{r_1}$))	$\mathbf{r} \sqsubseteq \mathbf{r_1}^-$
ObjectProperty($\mathbf{r}$ domain(C))	$\geqslant 1\mathbf{r} \sqsubseteq C$
ObjectProperty($\mathbf{r}$ range(C))	$\top \sqsubseteq \forall \mathbf{r}.C$
ObjectProperty($\mathbf{r}$ Functional)	$\top \sqsubseteq \leqslant 1\mathbf{r}$
ObjectProperty($\mathbf{r}$ InverseFunctional)	$\top \sqsubseteq \leqslant 1\mathbf{r}^-$
ObjectProperty($\mathbf{r}$ Symmetric)	$\mathbf{r} \sqsubseteq \mathbf{r}^-$
ObjectProperty($\mathbf{r}$ Transitive)	Trans($\mathbf{r}$)
EquivalentProperties($\mathbf{r_1}$ ... $\mathbf{r_n}$)	$\mathbf{r_i} \sqsubseteq \mathbf{r_j}, 1 \leq i; j \leq n$
SubPropertyOf($\mathbf{r_1}$ $\mathbf{r_2}$)	$\mathbf{r_1} \sqsubseteq \mathbf{r_2}$
Individual(o type(C))	$o : C$
Individual(o value($\mathbf{r}$ o_1))	$\langle o, o_1 \rangle : \mathbf{r}$
SameIndividual(o_1 ... o_n)	$o_1 = ... = o_n$
DifferentIndividuals(o_1 ... o_n)	$o_i \neq o_j, 1 \leq i < j \leq n$

Table 2.3: OWL-DL axioms and facts in DL abstract syntax

Axioms and facts in OWL-DL are analogous to axioms in description logics. Simplified overview (without loss of expressivity) of class axioms, property axioms and facts with correspondence to axioms in DL syntax are provided in Table 2.3. In this table, the same conventions are used as in Table 2.2. For more precise description of axioms and facts see OWL documentation [4, 7].

To preserve decidability of reasoning in OWL-DL, complex object properties cannot be defined as transitive. An object property is complex if either

1. it is specified as being functional or inverse-functional, or
2. there is some cardinality restriction that uses it, or
3. it has an inverse that is complex, or
4. it has a super-property that is complex.

3 Related work

When designing new ontology language, one has to take into account a trade-off between expressivity and tractability. Even relatively inexpressive language construct can cause serious computational issues in combination with others. Experiences from project DNAT revealed that support for n-ary relations is very important for annotating narratives. Some of the ontology languages, however, contains only binary relations and use so-called reification to replace n-ary ones. This chapter should give a reader overview on modern ontology languages, its way to solve problems with n-ary relations and its computational properties.

3.1 Conceptual Graphs

As a part of an European project CIPHER (Communities of Interest Promoting Heritage of European Regions), a document annotation tool DNAT (Dynamic Narrative Authoring Tool) was developed. It was designed to support users in creating printable knowledge-intensive content and the corresponding knowledge-base at the same time. Semantic annotation was visualized in Conceptual Graphs (CGs) while the knowledge model was a frame-based model of Apollo CH ontology editor [15] .

Conceptual graphs are human readable notation of First Order Logic (FOL), based on existential graphs [17] of Charles Sanders Pierce and semantic networks [16] of artificial intelligence. First paper on CGs, was published in 1976 by John F. Sowa who used them to represent the conceptual schemas in database systems. Main goal of CGs is to be able to model natural language as people understand it. In CGs it is very intuitive to model humans playing roles in events, capture knowledge about processes, methods, consequences of an action or event, etc. Since there exists a direct mapping from CGs to natural language, they can serve as an intermediate language for translating computer-oriented formalisms to and from natural language.

Experiments proved that CGs in DNAT were very well suited for annotating of documents. Authors of DNAT mentioned that most of the people learned to model knowledge in a few hours, since annotating process was very similar to using color markers for highlighting key concepts in documents. However, they realized that for further use of data, some more complex inference of annotated data was needed.

Semantic annotation difficulties

Since general CGs are equivalent to FOL, deduction is not decidable. This problem was partially solved by introducing simple conceptual graphs (SCGs). SCG is formally equivalent to the positive conjunctive existential fragment of FOL [14]. It means that they can express only conjunction of positive knowledge. The deduction problem can be performed by a graph homomorphism called projection, which is sound and complete. Intuitively, the existence of projection between two SCGs means that first graph is contained in the knowledge of second one. Main advantage of projection is that it can operate on pieces of user knowledge and visualize it in natural way as well. Although modelling knowledge in SCGs is very natural and intuitive, lack of negation and disjunction makes SCGs unusable for certain tasks. In addition to this problem, some work was done on adding negation and disjunction in limited way [18, 19]. Another inference techniques, instead of projection that compares two graphs by a global matching, used set of rules [20] to obtain derivation sequence from one graph to another. Similar expressivity problem of these approaches can be found in [21].

3.2 Web Ontology Language

Web Ontology Language (OWL) [4] is an expressive ontology language released as a W3C (World Wide Web Consortium) recommendation in February 2004. Multi-layered nature of language architecture provides Universal Resource Identifiers (URIs) to identify Web resources unambiguously via RDF [1] (basic assertional language) or defines classes and properties via RDFS [2] (schema language extension). In addition, OWL provide rich set of class constructors (e.g. logical combination of other classes, define value of class) and property constructors (e.g. property domain or range restriction, definition of superproperty). Since one of RDF notations is layered on top of XML [5], OWL comes with official exchange syntax RDF/XML [3]. Ontology terms can be linked across making it possible to cross-reference and reuse information. Also design based on the Web architecture, open (non-proprietary) nature of language, brought up OWL in front of all other ontology languages for Semantic Web.

Trade-off between expressivity and tractability, OWL addresses by defining three increasingly complex sublanguages: OWL Lite, OWL-DL, OWL Full. OWL Lite was defined with intention to provide minimal useful subset of language that support class and property classification hierarchies and simple constraint features. Tools for OWL Lite should be easier to implement, and thanks to good computational properties, it can be used to query large ontologies. On the other hand, OWL Lite lacks e.g. union and complement of concepts and cardinality restriction permits values of just 0 or 1.

Based on description logics (DL), the OWL-DL language was designed to support highest possible expressivity while retaining decidability. Originally, description logics (DL) focused on developing languages with polynomial-time complexity of reasoning problems. This way Knowledge Representation system based on DL would guarantee timely answers to the rest of the system. However, once very expressive DLs with exponential-time reasoning problems were implemented [26], it was recognized that knowledge bases of realistic size could be processed in reasonable time. Nowadays, many DLs have inference complexity ExpTime-hard or even worse. Complexity of OWL-DL entailment can be determined via correspondence to the satisfiability problem of SHOIN(D) description logic [28]. Unfortunately, most problems in SHOIN(D) including satisfiability of knowledge base, are known to be non-deterministic exponential time [27].

OWL Full would be useful for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

Decidability of OWL-Lite and OWL-DL helps ontology engineers to take advantage of reasoning services while constructing ontologies. For example, reasoner can check consistency of ontology or class satisfiability (if definition permits the class to have instances), construct class and property subsumption hierarchies, check subsumption, equivalency or disjointness of classes, list all individuals of a particular class and so on. Whenever an inconsistency occurs, techniques for error explanation or ontology repair [51] can be used. Some novel methods were implemented in ontology editor SWOOP [52].

Semantic annotation difficulties

Best candidate to fulfill both advanced annotation features and reasoning services at the same time is OWL-DL. Although OWL-DL is very expressive, there is no support for direct n-ary relation constructors. In July, 2004, W3C published first informative document about defining n-ary relations [6] in OWL. Document separates semantically different types of n-ary relations and provides appropriate design pattern for transformation using binary relations. Despite W3C's ambition, modelling n-ary relations met some negative consequences.

One of disadvantages is that W3C design patterns provide different representation choices for semantically same relation. The problem is caused by the polarity of the binary relation in RDF, i.e. distinction of a subject and an object of a triple. Since OWL is build on the top of RDF, every binary relation between two concepts in ontology must be build on subject-object basis. Despite the fact that even some obvious cases like a person related to a birthday might seem equally reasonable as a birthday related to a person, ontology modeller is forced to pick one representation. Considering n-ary relations, being represented by tree of concepts connected by binary relations, even much more different combinations exist.

Although W3C design patterns can relate n concepts, they don't capture all basic properties of n-ary relations. For example, let's have ternary relation birth that relates name of person, place and date of birth. It would be natural to expect that person with same name, place and date of birth cannot exist. To capture this constraint in OWL-DL is very counter-intuitive and complex problem and will be discussed in Chapter 5.

3.3 Nary Description Logics DLR

DLR is expressive logic formalism, whose intent is to extend DLs towards n-ary relations in natural way. Design of language was in inspired by ontology languages [34, 35, 36, 37] that had notion of n-ary relation already present. Distinguishing feature of DLR from any other DL formalism is capability of expressing inclusion axioms on complex relations. General inclusion axioms of relations are crucial for inter-schema assertions, as discussed in [34]. Also high expressivity of DLR is capable of capturing great variety of data models and knowledge representation formalism which makes it excellent choice for applications such as data integration.

As presented in [38], DLR can represent:

- the relational model, by considering only atomic relations and atomic concepts

- the entity relationship model in a straightforward way [37]
- an object-oriented data model, by restricting the use of existential and universal quantification in concept expressions, by restricting the attention to binary relations, and by eliminating negation and disjunction
- a "more traditional" DL, by restricting the attention to binary relations

Logical implication in DLR is decidable and EXPTIME-complete [39]. Practical, correct and complete algorithms exist in implemented systems and they are used in real applications [40, 41, 42, 43]. Thanks to expressivity and good computational properties, DLR seems to be interesting alternative for semantic annotation creation.

In many publications DLR was used for solving query containment problem (QCP). It is the problem of determining whether for every data (or knowledge) base, the result of one query is always a subset of the result of another query. The importance of this problem was addressed by both Database and Knowledge Representation community. Most of databases use schema expressed in a very simple data model (e.g. the relational model) and the query has the form of a conjunction of atomic queries (each involving one relation). On the contrary, current research on Knowledge Representation, typically assumes a more powerful schema specification language restricted to unary and binary predicates and query language where n-ary relations are either not used (e.g. queries as concepts) [10], or treated separately from the concepts and the roles of the schema (and therefore are not part of the knowledge base) [11] Calvanese et al. [39] proposed unified solution of query containment problem for both communities using theoretical framework based on logic DLR. High expressive power of DLR was used to formulate query and schema as well. Decision procedure of query containment was proved by a reduction to unsatisfiability of concept in knowledge base expressed in the EXPTIME-decidable DL CIQ [46].

Since there was no implementation of reasoner based on DL CIQ, Calvanese et al. solution did not lead to practical decidability. To overcome this problem, Horrocks et al. introduced slightly different mapping of query containment problem to DLR extended by ABox [45]. New encoding of containment problem [47] is much more natural (thanks to correspondence between variables of query and ABox individuals) and allows this problem to be reduced to a KB satisfiability problem in SHIQ DL. In both approaches resulting logics do not support n-ary relations, so they had to be reduced to binary ones by reification.

As explained in Chapter 2, every reasoning task in DLR can be transformed to satisfiability problem. If we would have algorithm for transforming DLR satisfiability to equivalent SHOIN DL satisfiability, we could use state-of-art reasoners to fully reason in DLR. Such an algorithm can be found as a part of solution of QCP in both approaches, however logics that DLR are transformed to are slightly different.

Transformation of DLR satisfiability to satisfiability of new KB can be divided into 2 steps. First, to ensure soundness of encoding, DLR is transformed by satisfiability-preserving translation function. For both approaches, this translation function is almost the same for TBox axioms, however differs in ABox axioms. Second, to ensure completeness of encoding, additional axioms are added so that new KB cannot become satisfiable if original DLR KB wasn't. In next subsections differences among both approaches will be revealed in more detailed way.

3.3.1 Reduction to CIQ DL satisfiability

Compared to SHOIN DL, CIQ DL lacks nominals, but contains qualified number restrictions and expressive role constructors, such as union, chaining and reflexive transitive closure of roles. Although CIQ DL contains *ABox* axioms, Calvanese at. al. decided to transform all DLR axioms to TBox axioms of new KB. They have introduced in new knowledge base so called sk-concepts for each DLR individual and representatives of tuple. Soundness of encoding for DLR *ABox* was provided by restrictions on sk-concepts. For completeness of encoding they needed to enforce that each sk-concept can be singled out (i.e. if there are two and more instances of sk-concept, they should represent same individual). It is done by additional axioms, using so called universal role (transitive closure of union of all roles).

3.3.2 Reduction to SHIQ DL satisfiability

Compared to SHOIN DL, SHIQ DL lacks nominals but contains qualified cardinality restriction. Soundness of encoding is more natural, compared to Calvanese at al., because DLR *ABox* axioms are transformed to similar SHIQ DL *ABox* axioms. For completeness of encoding, tuple-admissibility problem needed to be solved. It is done by additional axioms for each tuple occurring in DLR *ABox*. For this purpose, qualified number restrictions were used.

4 Nary Description Logics

Based on semantic annotation experiences from project DNAT, in this chapter, it is defined new Nary Description Logics $\mathcal{NDL}$. The logic $\mathcal{NDL}$ is based on Description Logics $\mathcal{DLR}$ without cardinality restrictions. $\mathcal{NDL}$ is enriched with new concept and relation constructors, which however present only syntactic sugar to the logic.

4.1 Syntax and Semantics

Syntax and semantics of $\mathcal{NDL}$ is based on $\mathcal{DLR}$ without cardinality restrictions. This section does not define $\mathcal{NDL}$ formally, but gives only brief overview for new constructors added to $\mathcal{NDL}$. For formal definitions of $\mathcal{DLR}$ see Section 2.4.1 and Section 2.4.2. In favor of semantic annotations new concept and relation constructors are added to $\mathcal{NDL}$. For quick preview of available constructors, see Figure 4.1. New concept constructors in $\mathcal{NDL}$ are *disjunction of concepts* $(C_1 \sqcup C_2)$ and *existential restriction* $(\exists[\$i]\mathbf{R})$. In addition, $\mathcal{NDL}$ has also relation constructor for *disjunction of relations* $(\mathbf{R_1} \sqcup \mathbf{R_2})$.

$\mathcal{NDL}$ concepts :	$\top,\ \neg C,\ C_1 \sqcap C_2,\ C_1 \sqcup C_2,\ \exists[\$i]\mathbf{R},\ \forall[\$i]\mathbf{R}$
$\mathcal{NDL}$ relations :	$\top_n,\ \neg\mathbf{R},\ \mathbf{R_1} \sqcap \mathbf{R_2},\ \mathbf{R_1} \sqcup \mathbf{R_2},\ (\$i/n : C)$

Figure 4.1: Syntax of $\mathcal{NDL}$

Semantics of $\mathcal{NDL}$ is summarized in Figure 4.2. Note that new constructors in $\mathcal{NDL}$ introduced only syntactic sugar to the logic : *disjunction of concepts* $C_1 \sqcup C_2$ can be equivalently represented as $\neg(C_1 \sqcap C_2)$; *disjunction of relations* $\mathbf{R_1} \sqcup \mathbf{R_2}$ is equivalent to $\neg(\mathbf{R_1} \sqcap \mathbf{R_2})$.

$$\begin{array}{rclrcl}
\top^{\mathcal{I}} &=& \Delta^{\mathcal{I}} & \top_n^{\mathcal{I}} &\subseteq& (\Delta^{\mathcal{I}})^n \\
C^{\mathcal{I}} &\subseteq& \top^{\mathcal{I}} & \mathbf{R}^{\mathcal{I}} &\subseteq& \top_n^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &=& \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} & (\neg \mathbf{R})^{\mathcal{I}} &=& \top_n^{\mathcal{I}} \backslash \mathbf{R}^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &=& C^{\mathcal{I}} \cap D^{\mathcal{I}} & (\mathbf{R} \sqcap \mathbf{S})^{\mathcal{I}} &=& \mathbf{R}^{\mathcal{I}} \cap \mathbf{S}^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &=& C^{\mathcal{I}} \cup D^{\mathcal{I}} & (\mathbf{R} \sqcup \mathbf{S})^{\mathcal{I}} &=& \mathbf{R}^{\mathcal{I}} \cup \mathbf{S}^{\mathcal{I}} \\
w^{\mathcal{I}} &\in& \top^{\mathcal{I}} & \boldsymbol{w}^{\mathcal{I}} &\in& \top_n^{\mathcal{I}}
\end{array}$$

$$\begin{aligned}
(\exists[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}}.d_i = d\} \\
(\forall[\$i]\mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \forall(d_1, ..., d_n) \in \mathbf{R}^{\mathcal{I}} \Rightarrow d_i = d\} \\
(\$i/n : C)^{\mathcal{I}} &= \{(d_1, ..., d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}
\end{aligned}$$

Figure 4.2: Semantics of $\mathcal{NDL}$

4.2 Graphical notation

To improve readability of $\mathcal{NDL}$ abstract syntax, we will use graphical notation, called display form (DF). DF is not meant to replace abstract syntax, but only show important structure of the Knowledge Base.

DF is oriented graph, that consist of two types of nodes: concept node and relation node. Each arc connect concept node with relation node. Concept nodes are drawn as rectangles. Relation nodes are drawn as circles, ovals, or ellipses. The arcs that link a relation node to a concept node are drawn as solid lines, preferably straight. To distinguish the arcs of an n-ary relation, an integer from 1 to n is drawn next to each arc.

Relation node represents $\mathcal{NDL}$ relation and is displayed as follows. Inside of the node is name of the $\mathcal{NDL}$ relation. From the node leads n outgoing arcs, where n is arity of relation. Concept node represent $\mathcal{NDL}$ concept. The concept name is displayed inside of the node and it is optionally followed by colon character ":" and individual name that has type of this concept. Other details of DF format should be self-explanatory from example in next section.

4.3 Example Knowledge Base

To clarify work in following chapters, we will use example of $\mathcal{NDL}$ Knowledge Base. Display form of this example KB is in Figure **??**. From the figure, it can be read : *DeidreCapron* is 18 years old *Person*, that *came* to *BletchleyPark* in $year1944$; $DeidreCapron$ $works$ in $BletchleyPark$ for 9 month. Schema of Deidre Knowledge Base in abstract syntax is in Figure 4.3. *ABox* axioms of Deidre KB in abstract syntax is in Figure 4.4.

$$\begin{aligned}
\textbf{hasAge} &\sqsubseteq (\$1/2 : Person) \sqcap (\$2/2 : NumberOfYears) \\
\textbf{come} &\sqsubseteq (\$1/3 : Person) \sqcap (\$2/3 : Place) \sqcap (\$3/3 : TimeSpec) \\
\textbf{work} &\sqsubseteq (\$1/3 : Person) \sqcap (\$2/3 : Place) \sqcap (\$3/3 : NumberOfMonths) \\
Year &\sqsubseteq TimeSpec \\
Person &\sqsubseteq \exists[\$1]\textbf{hasAge}
\end{aligned}$$

Figure 4.3: *TBox* axioms for Deidre example KB

$$\begin{aligned}
DeidreCapron &: Person \\
BletchleyPark &: Place \\
year1944 &: Year \\
\langle DeidreCapron, 18\rangle &: \textbf{hasAge} \\
\langle DeidreCapron, BletchleyPark, year1944\rangle &: \textbf{come} \\
\langle DeidreCapron, BletchleyPark, 9\rangle &: \textbf{work}
\end{aligned}$$

Figure 4.4: *ABox* axioms for Deidre example KB

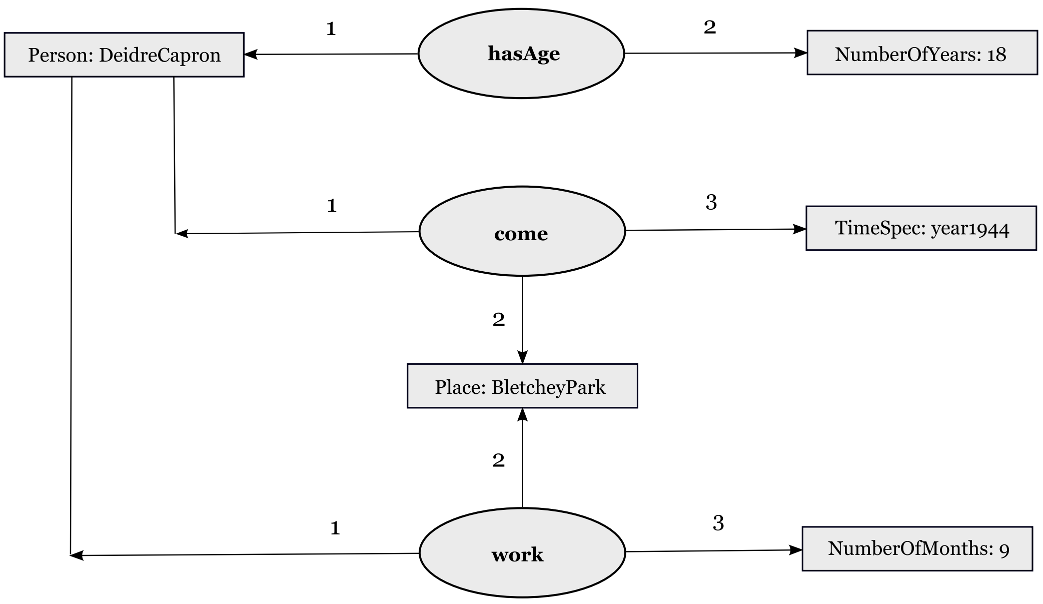

Figure 4.5: Display Form of Deidre example KB

5 Reification algorithms

In February 2004 W3C released recommendation for Web Ontology Language (OWL). From the three sublanguages defined in specification, the most interest was given to OWL-DL. OWL-DL as described in Chapter 2 has expressivity $\mathcal{SHOIN}$. Although OWL-DL is supported by most of the state-of-art semantic web tools, there are still some issues in implementation of reasoners. In particular, some of the reasoners cannot deal with *nominals*, some support them only approximately [55].

On the other hand, almost all of the state-of-art description logics reasoners can reason with *qualified cardinality restrictions*. Description logics based on OWL-DL without *nominals* and with *qualified cardinality restriction* is $\mathcal{SHIQ}$ DL. This description logics is part of new ontology language OWL-1.1, whose specification was released as W3C draft in January 2008.

Primary goal of this chapter is the design of reification algorithms for $\mathcal{NDL}$ according to work of Calvanese at al. [38] and Horrocks at al. [47]. The output binary logics of the algorithms should be supported by state-of-art description logics reasoners.

Since reification algorithm of Horrocks at al. is based on Calvanese at al. work, both algorithms share common characteristics that are discussed in Section 5.1. In the section also basic structure of algorithm and its proof is explained.

Based on Calvanese at al., new reification algorithm with output expressivity $\mathcal{SHIQ}$ is defined in Section 5.2. Same algorithm as defined in Horrocks at al. work, but only suited for $\mathcal{NDL}$, is defined in Section 5.3. The section also presents modification of this algorithm. The new algorithm reifies $\mathcal{NDL}$ into DL with expressivity $\mathcal{SHIF}$, which is subset of $\mathcal{SHIQ}$ without *cardinality restrictions*. Complications and contributions of this chapter is summarized in Section 5.4.

5.1 Common characteristics

Binary description logics, such as $\mathcal{SHOIQ}$ provide only unary predicates and binary relations. On the other hand, $\mathcal{DLR}$-like languages such as $\mathcal{NDL}$ allows for arbitrary n-ary relations. Thus, transformation of $\mathcal{NDL}$ into $\mathcal{SHOIQ}$ DL has to represent $\mathcal{NDL}$ n-ary relations by means of binary ones. This is done by process called *reification* [24].

Main idea behind *reification* is that each tuple of n-ary language is represented by one individual of binary language that is linked via dedicated functional relations to the elements of tuple. In following transformations roles $f_1, ..., f_n$ are used for this purpose. Distinguishing feature of Calvanese and Horrocks works compared to other approaches is realization of all properties of n-ary relations as understood in mathematical logic. To achieve this goal, *tuple-admissibility problem* of represented n-ary relations had to be fixed.

Tuple-admissibility is property of mathematical relations which ensures that relation is interpreted as set of tuples. Thus, if representation of relation by concept is *tuple-admissible*, there can exist only one instance of each tuple in the concept. Note that n-ary relations as defined by W3C for OWL-DL [6] are not *tuple-admissible.*

5.1.1 Structure of algorithm

The reification algorithm consists of two steps. First, a $\mathcal{NDL}$ KB is translated to binary DL by satisfiability-preserving translation $\sigma(.)$. Second, additional axioms are added to binary KB to guarantee that any model of new KB can be "un-reified" back into a model of the $\mathcal{NDL}$ KB. For this purpose function $f(.)$ is defined. Putting it all together, if $\mathcal{K}_{\mathcal{NDL}} = \langle \mathcal{S}, \mathcal{A} \rangle$ is an $\mathcal{NDL}$ KB, new KB will consist of axioms $\sigma(\mathcal{S}) \cup f(\mathcal{S}) \cup \sigma(\mathcal{A}) \cup f(\mathcal{A})$. To new (binary) KB it is referred in this chapter as to reified counterpart of $\mathcal{NDL}$ KB.

In both Calvanese and Horrocks works, reification of schema axioms is the same. However, the approaches differ in reification of *ABox* axioms. In Calvanese work *ABox* axioms were encoded by terminological (*TBox*) $\mathcal{SHIQ}$ axioms. Therefore further in text we will refer to the reification algorithm as T-SHIQ . On the other hand, in Horrocks work *ABox* axioms were encoded mainly by *ABox* axioms of reified counterpart. Thus further in text we will refer to the algorithms based on Horrocks as A-SHIQ and A-SHIF according to their expressivity. Moreover, translation functions according to Calvanese are labeled as σ' and f', while for Horrocks approach σ'' and f'' is used. All of the translation functions for *ABox* might be possibly subscripted.

5.1.2 Reification of schema

By translation function $\sigma(.)$, each atomic $\mathcal{NDL}$ relation is represented by a new atomic concept, each $\mathcal{NDL}$ relation inclusion axiom is translated to a concept inclusion axiom.

Definition 5.1.1. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base. The reification translation function $\sigma(.)$ for TBox is defined as follows:*

$$\begin{aligned} \sigma(\top) &= \top_1 \\ \sigma(A) &= A \\ \sigma(\neg C) &= \neg\sigma(C) \\ \sigma(C_1 \sqcap C_2) &= \sigma(C_1) \sqcap \sigma(C_2) \\ \sigma(C_1 \sqcup C_2) &= \sigma(C_1) \sqcup \sigma(C_2) \\ \sigma(\exists[\$i]\mathbf{R}) &= \exists f_i^-.\sigma(\mathbf{R}) \\ \sigma(\forall[\$i]\mathbf{R}) &= \forall f_i^-.\sigma(\mathbf{R}) \end{aligned} \qquad \begin{aligned} \sigma(\top_n) &= \top_n \\ \sigma(\mathbf{P}) &= A_\mathbf{P} \\ \sigma(\neg\mathbf{R}) &= \top_n \sqcap \neg\sigma(\mathbf{R}) \\ \sigma(\mathbf{R_1} \sqcap \mathbf{R_2}) &= \sigma(\mathbf{R_1}) \sqcap \sigma(\mathbf{R_2}) \\ \sigma(\mathbf{R_1} \sqcup \mathbf{R_2}) &= \sigma(\mathbf{R_1}) \sqcup \sigma(\mathbf{R_2}) \\ \sigma(\$i/n : C) &= \top_n \sqcap \exists f_i.\sigma(C) \end{aligned} \tag{5.1}$$

$$\begin{aligned} \sigma(\mathcal{S}) = &\{(\sigma(C_1) \sqsubseteq \sigma(C_2)) \mid (C_1 \sqsubseteq C_2) \in \mathcal{S}\} \\ &\cup \{(\sigma(\mathbf{R_1}) \sqsubseteq \sigma(\mathbf{R_2})) \mid (\mathbf{R_1} \sqsubseteq \mathbf{R_2}) \in \mathcal{S}\} \end{aligned} \tag{5.2}$$

$A_\mathbf{P}$, A, $\top_1$,..., $\top_{n_{max}}$ are newly introduced atomic concepts, f_1,..., $f_{n_{max}}$ newly introduced atomic roles and n_{max} denote maximum arity of relations appearing in $\mathcal{NDL}$ KB. In the following definition function $f(.)$ for $\mathcal{NDL}$ schema is defined. The function provides additional axioms to reified counterpart that are needed for "un-reification".

Definition 5.1.2. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base. $f(\mathbf{S})$ is a set consisting of following axioms (where $x \equiv y$ is an abbreviation of $x \sqsubseteq y$ and $y \sqsubseteq x$):*

$$\top \sqsubseteq \top_1 \sqcup ... \sqcup \top_{n_{max}} \tag{5.3}$$

$$\top \sqsubseteq (\leq 1 f_1) \sqcap ... \sqcap (\leq 1 f_{n_{max}}) \tag{5.4}$$

$$\forall f_i.\bot \sqsubseteq \forall f_{i+1}.\bot \qquad \textit{for } 1 \leq i < n_{max} \tag{5.5}$$

$$T_i \equiv \exists f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot \qquad \textit{for } 2 \leq i \leq n_{max} \tag{5.6}$$

$$A \sqsubseteq \top_1 \qquad \textit{for each atomic concept } A \tag{5.7}$$

$$A_\mathbf{P} \sqsubseteq \top_n \qquad \textit{for each atomic relation } \mathbf{P} \textit{ of arity } n \tag{5.8}$$

Intuitively, 5.3 states, that each individual has to belong to at least one of T_i for $1 \leq i \leq n_{max}$. T_1 can be viewed as set of simple individuals, while elements of T_i for $i \geq 2$ can be interpreted as instances of tuples of arity i. Note that in combination with 5.6, it can be proved that T_i for $i \geq 2$ are mutually disjoint. Roles $f_1, ..., f_n$ are used to link an instance of tuple to its elements. Axiom 5.4 states that each f_i is functional. Axiom 5.5 ensures, that each individual that is connected with f_i must be connected also with all f_j, for $1 \leq j < i$. Structure of concepts that hold all instances of tuples of one arity is defined by axioms 5.6. By axioms 5.7 and 5.8, it is defined that each unary concept is subset of T_1 and each concept representative for n-ary relation is subset of appropriate T_i.

Example 5.1.1. *To illustrate reification of schema axioms consider example knowledge base about Deidre Capron from Chapter 4. If $\mathcal{S}$ is schema of Deidre Capron KB than $\sigma(\mathcal{S}) \cup f(\mathcal{S})$ is following set of axioms:*

$$\begin{array}{rlrl}
A_{come} \sqsubseteq & (\top_3 \sqcap \exists \mathbf{f_1}.Person) \sqcap & A_{work} \sqsubseteq & (\top_3 \sqcap \exists \mathbf{f_1}.Person) \sqcap \\
& (\top_3 \sqcap \exists \mathbf{f_2}.Place) \sqcap & & (\top_3 \sqcap \exists \mathbf{f_2}.Place) \sqcap \\
& (\top_3 \sqcap \exists \mathbf{f_3}.TimeSpec) & & (\top_3 \sqcap \exists \mathbf{f_3}.NoMonths) \\
A_{hasAge} \sqsubseteq & (\top_2 \sqcap \exists \mathbf{f_1}.Person) \sqcap & Year \sqsubseteq & TimeSpec \\
& (\top_2 \sqcap \exists \mathbf{f_2}.NoYears) & Person \sqsubseteq & \exists \mathbf{f_1^-}.A_{hasAge}
\end{array}$$

$$\begin{array}{rl}
T \sqsubseteq & T_1 \sqcap T_2 \sqcap T_3 \\
T \sqsubseteq & (\leq 1\mathbf{f_1}) \sqcap (\leq 1\mathbf{f_2}) \sqcap (\leq 1\mathbf{f_3})
\end{array}$$

$$\begin{array}{rlrl}
\forall \mathbf{f_1}.\bot \sqsubseteq & \forall \mathbf{f_2}.\bot & \top_2 \equiv & \exists \mathbf{f_1} \top_1 \sqcap \exists \mathbf{f_2} \top_1 \sqcap \forall \mathbf{f_3} \bot \\
\forall \mathbf{f_2}.\bot \sqsubseteq & \forall \mathbf{f_3}.\bot & \top_3 \equiv & \exists \mathbf{f_1} \top_1 \sqcap \exists \mathbf{f_2} \top_1 \sqcap \exists \mathbf{f_3} \top_1
\end{array}$$

$$\begin{array}{rrr}
Place \sqsubseteq \top_1 & NoYears \sqsubseteq \top_1 & Person \sqsubseteq \top_1 \\
TimeSpec \sqsubseteq \top_1 & NoMonths \sqsubseteq \top_1 & Year \sqsubseteq \top_1 \\
A_{hasAge} \sqsubseteq \top_2 & A_{work} \sqsubseteq \top_3 & A_{come} \sqsubseteq \top_3
\end{array}$$

In the following sections, two different types of reifications of $\mathcal{NDL}$ into binary DLs are described. Main difference of approaches are the way how they treat encoding of individuals. The first approach based on Calvanese work encodes $\mathcal{NDL}$ individuals assertions into terminology axioms, while the second one uses mostly *ABox* assertions instead.

5.2 Reification using TBox

According to the Calvanese work, this section defines reified counterpart of $\mathcal{NDL}$ *ABox*, where each individual and tuple is represented by special concept. *ABox* assertions are then represented as terminological axioms of reified counterpart. Since each representative concept is interpreted as a set of individuals, some additional axioms need to be added to the KB to ensure that each representative concept can be interpreted as a single individual. Singling out of concepts is done through so called universal accessibility role. Same technique is also used for representative concepts of tuples in order to gain tuple-admissible model. The reification algorithm outputs a binary KB with expressivity $\mathcal{SHIQ}$.

5.2.1 Representation of individuals and tuples by concepts

Reification algorithm according to Calvanese at al. represents each $\mathcal{NDL}$ individual w by new atomic concept A_w and each $\mathcal{NDL}$ tuple $\vec{w}$ by new atomic concept $T_{\vec{w}}$. Individual and tuple assertions of original $\mathcal{NDL}$ KB are represented as inclusion assertions in reified counterpart involving such new atomic concepts. However the new concepts alone do not suffice to represent faithfully the original $\mathcal{NDL}$ KB, because each individual w (tuple $\vec{w}$) of original KB is represented by a set of individuals of A_w ($T_{\vec{w}}$) in reified counterpart. In order to relate the satisfiability of the $\mathcal{NDL}$ KB to the satisfiability of its reified counterpart, we must be able to find a model where each of representative concepts are interpreted by single individuals for each model of the reified counterpart. For this purpose, a role U is defined and following concept inclusion axioms are added to reified counterpart for each representative concept A and some concepts C:

$$(A \sqcap C) \sqsubseteq \forall U.(\neg A \sqcup C)$$

Let's assume that U is interpreted as a role that relates every two individuals of reified counterpart to each other. Thus, the above axiom states that if an individual of A is also a member of C, than every individual of A is also a member of C. Observe that if we could add infinite set of axioms of this form to the reified counterpart for each possible concept of the language, there would be no way in the logic to distinguish two individuals of each A one from the another. Hence, we could safely restrict models of reified counterpart to models that interpret each representative concept A by only one individual. Moreover, as showed in [46], for this purpose a finite set of concepts C is sufficient. Based on this work it is defined extension of Fischer-Ladner closure $CL'(.)$.

In the following text two main complication of adapting Calvanese work to $\mathcal{SHIQ}$ DL are discussed. The First is the construction of an universal accessibility role. Second, we have to extend the notion of the Fischer-Ladner closure $CL(.)$.

5.2.2 Construction of the Universal accessibility role

Universal accessibility role is defined as cartesian cross-product of all individuals in the interpretation domain, i.e. for $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Due to connected-model property[1] of underlying logics $\mathcal{CIQ}$ DL, Calvanase at. al. [38] could represent U as transitive closure of union of all roles and their inverses. $\mathcal{SHIQ}$ DL has connected-model property, however it lacks union constructors on roles. Thus, we define pseudo-universal role U that is superset of universal accessibility role and prove that it is sufficient for our purpose.

Definition 5.2.1 (Construction of pseudo-universal role)**.** *Let $\mathcal{R}$ be set of atomic roles, than $UC(R)$ is set of axioms that construct pseudo-universal role U as follows:*

$$UC(\mathcal{R}) = \{p \sqsubseteq U, p^{-} \sqsubseteq U \mid p \in \mathcal{R}\} \cup \{trans(U)\} \tag{5.9}$$

Note that 5.9 defines U for which $(\bigsqcup_{\mathcal{R}} r)^{*} \sqsubseteq U$ holds.

5.2.3 Construction of the extended Fischer-Ladner closure

Set of concepts that needs to be used in axioms for singling out representative concepts is finite. It will be defined by an extension of the Fischer-Ladner closure. Definition of original Fischer-Ladner closure modified for $\mathcal{SHIQ}$ DL concept follows.

Definition 5.2.2. *[Fischer-Ladner closure for $\mathcal{SHIQ}$ DL concept] Let C_0 be $\mathcal{SHIQ}$ DL concept, the Fischer-Ladner closure $CL(C_0)$ of concept C_0, is defined as the smallest set of concepts such that $C_0 \in CL(C_0)$ and such that (assuming $\sqcup$ and $\forall$ to be expressed by means of $\sqcap$ and $\exists$, and the inverse operator applied only to atomic roles)[49]:*

[1] if a formula (an axiom) has a model, than it has one that is connected when viewing it as a graph.

if $C \in CL(C_0)$	*then* $\neg C \in CL(C_0)$ *(if* C *is not of the form* $\neg C'$*)*
if $\neg C \in CL(C_0)$	*then* $C \in CL(C_0)$
if $C \sqcap C' \in CL(C_0)$	*then* $C, C' \in CL(C_0)$
if $\exists \mathbf{r}.C \in CL(C_0)$	*then* $C \in CL(C_0)$

Intuitively, $CL(C)$ is analogous to the set of subconcepts in simpler logics: It comprises the concepts that play direct role in establishing the interpretation of C. The size of $CL(C)$ is linearly bounded by the size of C [49]. By the definition, if $C' \in CL(C)$, then $CL(C') \subseteq CL(C)$.

Extension of Fischer-Ladner closure towards the whole knowledge base is created as follows.

Definition 5.2.3 (Extended Fischer-Ladner closure for schema axioms of reified counterpart)**.** *Let $\mathcal{S}$ be a set of terminological axioms of the reified counterpart and $\{A_1, ..., A_m\}$ is a set of all concept representatives used in the reification algorithm; extended Fischer-Ladner closure $CL'(\mathcal{S})$ of the schema $\mathcal{S}$ is defined as smallest set of concepts such that:*

$C \in CL'(\mathcal{S})$	*if* $C \in CL(C_{\mathcal{S}})$
$\exists \mathbf{r}.((\neg A_1 \sqcap ... \sqcap \neg A_m) \sqcap C) \in CL'(\mathcal{S})$	*if* $\exists \mathbf{r}.C \in CL(C_{\mathcal{S}})$
$\exists \mathbf{r}.A_i \in CL'(\mathcal{S})$	*for each* $\mathbf{r}$ *and* $i \in \{1, ..., m\}$
$\exists \mathbf{r}.\bot \in CL'(\mathcal{S})$	*for each* $\mathbf{r}$

where

$$C_{\mathcal{S}} = \bigsqcap_{A \sqsubseteq B \in \mathcal{S}} (\neg A \sqcup B)$$

Note that this is the same definition as in [46], however some construct from $\mathcal{CIQ}$ DL were substituted by equivalent construct of $\mathcal{SHIQ}$ DL (e.g. concepts like $\exists(f \circ id(C)).D$ were substituted by $\exists f.(C \sqcap D)$).

5.2.4 Transformation of the *ABox*

Reified counterpart according to Calvanese at al. work is constructed as follows.

Construction 5.2.1. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base, $\mathcal{U}_{\mathcal{A}}$ is set of all individuals occurring in $\mathcal{A}$, and $\mathcal{W}_{\mathcal{A}}$ is set of all tuples occurring in $\mathcal{A}$. The reified counterpart $\wp(\mathcal{K})$ of $\mathcal{K}$ is constructed as follows:*

$$\begin{aligned}
\sigma'(w : C) &:= \{A_w \sqsubseteq \sigma(C)\} \\
\sigma'(\vec{w} : \mathbf{R}) &:= \{T_{\vec{w}} \sqsubseteq \sigma(\mathbf{R})\} \\
&\quad \cup \{T_{\vec{w}} \equiv \exists f_1.A_{w_1} \sqcap ... \sqcap \exists f_n.A_{w_n} \sqcap \forall f_{n+1}.\bot\} \\
&\quad \cup \{A_{w_i} \sqsubseteq \exists f_i^-.T_{\vec{w}} \sqcup \leq 1 f_i^-.T_{\vec{w}}\}
\end{aligned}$$

Translation function $\sigma'(.)$ for $\mathcal{NDL}$ ABox is then defined as follows:

$$\sigma'(\mathcal{A}) \quad := \quad \bigcup\{\sigma'(A) \mid A \in \mathcal{A}\}$$

In order to gain expected behavior of representative concepts functions $f_1'(\mathcal{A})$ and $f_2'(\mathcal{A})$ are constructed. The set of axioms $f_1'(\mathcal{A})$ contains only one assertion that ensures that all representative concepts are satisfiable, i.e. contains at least one individual. For this purpose, newly introduced role create and individual s_{root} is used:

$$f_1'(\mathcal{A}) \quad := \quad \{\ s_{root} : \exists create.A_{w_1} \sqcap ... \sqcap \exists create.A_{w_m} \mid \{w_1, ..., w_m\} = \mathcal{U}_{\mathcal{A}} \cup \mathcal{W}_{\mathcal{A}}\}$$

Let $CL'(.)$ be extended fisher-ladner closure as described in Definition 5.2.3, and $UC(.)$ set of axioms that construct pseudo-universal role U as described in Definition 5.2.1. If $\mathcal{R}_{\mathcal{M}}$ abbreviate set of atomic roles occurring in set of axioms $\mathcal{M}$, and $\mathcal{A}_{\mathcal{M}}$ abbreviate set of all individual and tuple representatives used in set of axioms $\mathcal{M}$ than function $f_2'(\mathcal{A})$ is constructed as follows:

$$\begin{aligned}
\mathcal{S}' \quad &:= \quad \sigma(\mathcal{S}) \cup f(\mathcal{S}) \cup \sigma'(\mathcal{A}) \\
f_2'(\mathcal{A}) \quad &:= \quad UC(\mathcal{R}_{\mathcal{S}'} \cup \{create\}) \\
&\quad\quad \cup \{\ A \sqcap C \sqsubseteq \forall U.(\neg A \sqcup C) \mid A \in \mathcal{A}_{\mathcal{S}'}, C \in CL'(\mathcal{S}')\}
\end{aligned}$$

The set of axioms $f_2'(\mathcal{A})$ ensures that all representative concepts are singled out. Finally, the reified counterpart $\wp(\mathcal{K})$ is defined as follows:

$$\wp(\mathcal{K}) \quad := \quad \langle \sigma(\mathcal{S}) \cup f(\mathcal{S}) \cup \sigma'(\mathcal{A}) \cup f_2'(\mathcal{A}), f_1'(\mathcal{A})\rangle$$

□

5.2.5 Correctness of algorithm

Lemma 1. *(Soundness of encoding) Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A}\rangle$ be a $\mathcal{NDL}$ knowledge base and $\wp(\mathcal{K})$ its reified counterpart according to TBox reification algorithm. If $\mathcal{K}$ is satisfiable, then $\wp(\mathcal{K})$ is satisfiable.*

Proof. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of $\mathcal{K}$. From $\mathcal{I}$, an interpretation $\mathcal{I}' := (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ of $\wp(\mathcal{K})$ can be build as follows:

1. $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{s_{root}\} \cup \bigcup_{n \in \{2,...,n_{max}\}} \{s_{\vec{d}} \mid \vec{d} \in \top_n^{\mathcal{I}}\}$;

2. $\top_n^{\mathcal{I}'} = \{s_{(d_1,...,d_n)} \mid (d_1, ..., d_n) \in T_n^{\mathcal{I}}\}$, for each $n \in \{2, ..., n_{max}\}$;

3. $f_i^{\mathcal{I}'} = \{(s_{(d_1,...,d_n)}, d_i) \mid (d_1, ..., d_n) \in \top_n^{\mathcal{I}}\}$, for each $n \in \{2, ..., n_{max}\}$ and $i \in \{2, ..., n\}$;

4. $A_{\mathbf{P}}^{\mathcal{I}'} = \{s_{(d_1,...,d_n)} \mid (d_1, ..., d_n) \in \mathbf{P}^{\mathcal{I}}\}$, for each atomic relation $\mathbf{P}$;

5. $\top_1^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{s_{root}\}$;

6. $A^{\mathcal{I}'} = A^{\mathcal{I}}$, for each atomic concept A;

7. $A_w^{\mathcal{I}'} = \{d\}$, for the element domain $d = w^{\mathcal{I}}$;

8. $T_{\vec{w}}^{\mathcal{I}'} = \{s_{\vec{d}}\}$, for the tuple $\vec{d} = \vec{w}^{\mathcal{I}}$ of element domains;

9. $create^{\mathcal{I}'} = \{(s_{root}, s) \mid s \in C_R,\ for\ some\ representative\ concept\ C_R\}$

It is easy to verify that constructed interpretation $\mathcal{I}'$ is model of $\wp(\mathcal{K})$. $\square$

Definition 5.2.4. *A model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\wp(\mathcal{K})$ is admissible if it is tuple-admissible and each representative concept is satisfied by exactly one individual. If $B_1, ..., B_K$ are denoted to all representative concepts of $\wp(\mathcal{K})$, than admissible model $\mathcal{I}$ is also a pseudo-tree admissible model if it has the following form:*

- $\exists s_{B_1}, ..., s_{B_K} \in \Delta^{\mathcal{I}} \forall i \in \{1, ..., K\} : B_i^{\mathcal{I}} = \{s_{B_i}\}$
- $\exists s_{root} : create^{\mathcal{I}} = \{(s_{root}, s_{B_i}) \mid i \in \{1, ..., K\}\}$
- *each maximal component of $\Delta^{\mathcal{I}} \setminus (\{s_{root}\} \cup \{s_{B_i} \mid i \in \{1, ..., K\}\})$ is a tree, when viewed as an undirected graph*

Lemma 2. *Let $\wp(\mathcal{K})$ be reified counterpart of some $\mathcal{NDL}$ knowledge base $\mathcal{K}$ according to TBox reification algorithm. If $\wp(\mathcal{K})$ is satisfiable, then it has pseudo-tree admissible model.*

Proof. If $\wp(\mathcal{K})$ is satisfiable, than by *tree-model property*[2] of $\mathcal{SHIQ}$, $\wp(\mathcal{K})$ admits tree-model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ rooted by the only asserted individual of $\wp(\mathcal{K})$. From the tree-structure of the model, it is obvious that there is no pair of individuals that represent the same reified tuple and therefore $\mathcal{I}$ is tuple-admissible.

[2]if a formula (an axiom) has a model, than it is satisfiable at the root of a model based on a tree. Note that it is even stronger condition than connected-model property.

Pseudo-tree admissible model $\mathcal{I}'' = (\Delta^{\mathcal{I}''}, \cdot^{\mathcal{I}''})$ with $\Delta^{\mathcal{I}''} \subseteq \Delta^{\mathcal{I}}$ is obtained from $\mathcal{I}$ in these two steps. First, a model $\mathcal{I}' = (\Delta^{\mathcal{I}''}, \cdot^{\mathcal{I}'})$ is constructed by modifying tree-structure of $\mathcal{I}$. Second, $\mathcal{I}''$ is defined by interpreting each representative concept of $\mathcal{I}'$ as a singleton.

Let $s_{root} \in \Delta^{\mathcal{I}}$ be the root of $\mathcal{I}$. We choose one individual s_{B_i} from each representative concept B_i such that $(s_{root}, s_{B_i}) \in create^{\mathcal{I}}$. Model $\mathcal{I}'$ is defined as follows:

1. $create^{\mathcal{I}'} = \{(s_{root}, s_{B_i}) \in create^{\mathcal{I}} \mid i \in \{1, ..., K\}\}$
2. $f^{\mathcal{I}'} = (f^{\mathcal{I}} \backslash (\{(s_{B_i}, s) \in f^{\mathcal{I}} \mid s \in B_j^{\mathcal{I}}, i, j \in \{1, ..., K\}\} \cup$
   $\{(s, s_{B_j}) \in f^{\mathcal{I}} \mid s \in B_i^{\mathcal{I}}, i, j \in \{1, ..., K\}\}))$
   $\cup \{(s_{B_i}, s_{B_j}) \mid (s_{B_i}, s) \in f^{\mathcal{I}}, s \in B_j^{\mathcal{I}}, i, j \in \{1, ..., K\}\})$
   for each atomic role f except *create* and U
3. $U^{\mathcal{I}'} = U^{\mathcal{I}} \cap (\Delta^{\mathcal{I}'} \times \Delta^{\mathcal{I}'})$
4. $A^{\mathcal{I}'} = A^{\mathcal{I}} \cap \Delta^{\mathcal{I}'}$ for each atomic concept A including representative concepts
5. $\Delta^{\mathcal{I}'} = \{s_{root}\} \cup \{s \in \Delta^{\mathcal{I}'} \mid (s_{root}, s) \in create^{\mathcal{I}'} \circ (\bigcup_f (f^{\mathcal{I}'} \cup (f^-)^{\mathcal{I}'}))^*\}$

Construction above chooses one individual s_{B_i} from each representative concept B_i among all that are connected to s_{root} through role *create* (1). Let B_i and B_j be some representative concepts related through role f such that B_i represents tuple and B_j its component. From axiom (5.4), s_{B_i} can have only one f-successor s_j, which is instance of B_j. Symmetrically, from construction of σ' for *ABox* relation assertions, s_{B_j} can have only one f-predecessor s_i, which is instance of B_i. In (2), s_{B_i} and s_{B_j} are disconnected from s_i and s_j respectively and connected via f. Subtrees rooted at s_i and s_j are then cut away by (5), which defines interpretation domain as maximally connected component of the Kripke structure that includes s_{root}. Notice, that number of incoming and outgoing f-edges of s_{B_i} and s_{B_j} did not change.

Let $\mathcal{S}'$ be set of axiom as denoted by construction in 5.2.1, and $C_{\mathcal{S}'} = \bigsqcap_{C \sqsubseteq D \in \mathcal{S}'} \neg C \sqcup D$. By (3), U is restricted to represent exactly transitive-closure of all roles and its inverses and therefore $\mathcal{I}'$ can be turned in a straightforward way into a model of the $\mathcal{CIQ}$ KB. In a $\mathcal{CIQ}$ KB by the construction in Lemma 5 of [46], it is possible to show that for each concept $C \in CL(C_{\mathcal{S}'}))$ and for each individual $s \in \Delta^{\mathcal{I}'}$, we have $s \in C^{\mathcal{I}'}$ if and only if $s \in C^{\mathcal{I}'}$. Hence, since $C_{\mathcal{S}'} \in CL(C_{\mathcal{S}'})$, we get that $s_{root} \in \mathcal{I}'$.

Model $\mathcal{I}''$ is constructed form $\mathcal{I}'$ by interpreting each representative concept as singleton, thus $B_i = \{s_{B_i}\}$ for each $i \in \{1, ..., K\}$. □

Theorem 1. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base and $\wp(\mathcal{K})$ its reified counterpart according to TBox reification algorithm. Then $\mathcal{K}$ is satisfiable, if and only if $\wp(\mathcal{K})$ is satisfiable.*

Proof. The only if direction is proved by Lemma 1.

For the converse direction from Lemma 2, let $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be *pseudo-tree admissible model.* An interpretation $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ can be build from $\mathcal{I}'$ as follows:

1. $\Delta^{\mathcal{I}} = \top_1^{\mathcal{I}'}$;
2. $\mathbf{P}^{\mathcal{I}} = \{(s_1, ..., s_n) \mid \exists s' \in A_{\mathbf{P}}^{\mathcal{I}'}.((s', s_i) \in f_i^{\mathcal{I}'}, \text{for } i \in \{1, ..., n\})\}$, for each relation representative $A_{\mathbf{P}}$ included in $\top_n$;
3. $A^{\mathcal{I}} = A^{\mathcal{I}'}$, for each atomic concept A included in $\top_1$;
4. $w^{\mathcal{I}} = w^{\mathcal{I}'}$, for each individual w from $\top_1$

It is easy to verify that constructed interpretation $\mathcal{I}$ is model of $\mathcal{K}$. □

5.3 Reification using *ABox*

This section defines two different algorithms for reification of $\mathcal{NDL}$ *ABox* based on Horrocks at. al work. In algorithms, both individuals and tuples are encoded by individuals. $\mathcal{NDL}$ concept assertions are converted to concept assertions of the reified counterpart. $\mathcal{NDL}$ relation assertions of arity n are reified into n role assertions and one concept assertion.

Both algorithms differs in the way of solving the tuple-admissibility problem. The first algorithm is identical to one defined by Horrocks in [47], but suited for $\mathcal{NDL}$ KB. To ensure tuple-admissibility of relations, it defines one concept assertion for each tuple. Expressivity of reified counterpart is $\mathcal{SHIQ}$
.

The second algorithm addresses tuple-admissibility problem by creating a new role and adding one general concept inclusion axiom for each tuple. Reified counterpart of the algorithm has the expressivity $\mathcal{SHIF}$.

Construction of reified counterpart for the first algorithm is done as follows.

Construction 5.3.1. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base. The reified counterpart $\phi(\mathcal{K})$ of $\mathcal{K}$ is constructed as follows:*

$$\begin{aligned}\sigma''(w : C) =& \{w : \sigma(C)\}\\ \sigma''(\vec{w} : \mathbf{R}) =& \{t_{\vec{w}} : \sigma(\mathbf{R})\} \cup \{\langle t_{\vec{w}}, w_i\rangle : f_i \mid 1 \leq i \leq n\}\end{aligned}$$

Translation function $\sigma''(.)$ for $\mathcal{NDL}$ ABox is then defined as follows:

$$\sigma''(\mathcal{A}) \quad := \quad \bigcup\{\sigma''(A) \mid A \in \mathcal{A}\}$$

If $\mathcal{U}_\mathcal{A}$ is set of all individuals occurring in $\mathcal{A}$ and $\mathcal{W}_\mathcal{A}$ is set of all tuples occurring in $\mathcal{A}$, the set of axioms $f''(A)$ that ensures tuple-admissibility of asserted tuples is defined as follows:

$$\begin{aligned}f''(\mathcal{A}) =& \{w : Q_w \mid w \in \mathcal{U}_\mathcal{A}\} \cup\\ & \{w_1 :\leq f_1^-(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap \ldots \sqcap \exists f_n.Q_{w_n}) \mid \vec{w} = \langle w_1, \ldots, w_n\rangle \in \mathcal{W}_\mathcal{A}\}\end{aligned}$$

The reified counterpart $\phi(K)$ is defined as follows:

$$\phi(\mathcal{K}) \quad := \quad \langle \sigma(\mathcal{S}) \cup f(\mathcal{S}), \sigma''(\mathcal{A}) \cup f''(\mathcal{A})\rangle$$

□

5.3.1 Correctness of reification algorithms

Lemma 3. *(Soundness of encoding) Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A}\rangle$ be a $\mathcal{NDL}$ knowledge base and $\phi(\mathcal{K})$ its reified counterpart according to the ABox reification algorithm. If $\mathcal{K}$ is satisfiable, then $\phi(\mathcal{K})$ is satisfiable.*

Proof. Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be a model of $\mathcal{K}$. From $\mathcal{I}$, an interpretation $\mathcal{I}' := (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ of $\wp(\mathcal{K})$ can be build as follows:

1. $\Delta^{\mathcal{I}'} = \Delta^\mathcal{I} \cup \bigcup_{n \in \{2,\ldots,n_{max}\}}\{t_{\vec{d}} \mid \vec{d} \in \top_n^\mathcal{I}\}$;
2. $\top_n^{\mathcal{I}'} = \{t_{(d_1,\ldots,d_n)} \mid (d_1, \ldots, d_n) \in T_n^\mathcal{I}\}$, for each $n \in \{2, \ldots, n_{max}\}$;
3. $f_i^{\mathcal{I}'} = \{(t_{(d_1,\ldots,d_n)}, d_i) \mid (d_1, \ldots, d_n) \in \top_n^\mathcal{I}\}$, for each $n \in \{2, \ldots, n_{max}\}$ and $i \in \{2, \ldots, n\}$;
4. $A_\mathbf{P}^{\mathcal{I}'} = \{t_{(d_1,\ldots,d_n)} \mid (d_1, \ldots, d_n) \in \mathbf{P}^\mathcal{I}\}$, for each atomic relation $\mathbf{P}$;

5. $\top_1^{\mathcal{I}'} = \Delta^{\mathcal{I}}$;
6. $A^{\mathcal{I}'} = A^{\mathcal{I}}$, for each atomic concept A;
7. $Q_w^{\mathcal{I}'} = \{w^{\mathcal{I}}\ \}$, for each representative concept Q_w;
8. $w^{\mathcal{I}'} = d$, for the element domain $d = w^{\mathcal{I}}$;
9. $t_{\vec{w}}^{\mathcal{I}'} = \{t_{\vec{d}}\}$, for the tuple $\vec{d} = \vec{w}^{\mathcal{I}}$ of element domains;

It is easy to verify that the constructed interpretation $\mathcal{I}'$ is the model of $\phi(\mathcal{K})$.

□

Definition 5.3.1. *(tuple-admissible model) Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A}\rangle$ be $\mathcal{SHIQ}$ knowledge base that is the reified counterpart of some $\mathcal{NDL}$ knowledge base. Model $\mathcal{I}$ of $\mathcal{K}$ is tuple-admissible if the following condition holds:*

$$(\bigwedge_{1\le i\le n} \exists d.(\langle t, d\rangle \in f_i^{\mathcal{I}} \wedge \langle t', d\rangle \in f_i^{\mathcal{I}})) \Rightarrow t = t' \tag{5.10}$$

If there are tuple representatives that violates condition 5.10, we will refer to them as *conflicting tuple representatives.*

Lemma 4. *(Tuple-admissibility of asserted individuals) Let $\phi(\mathcal{K})$ be the reified counterpart of some $\mathcal{NDL}$ knowledge base $\mathcal{K}$ according to the ABox reification algorithm. If $\phi(\mathcal{K})$ satisfiable, than there is no conflict between any two tuple representatives of ABox.*

Proof. Let $\mathcal{I}$ be model of $\phi(\mathcal{K})$ and $t_{\vec{w}}^{\mathcal{I}}$, $t_{\vec{v}}^{\mathcal{I}}$ two individuals of *ABox*. If $t_{\vec{v}}^{\mathcal{I}} = t_{\vec{w}}^{\mathcal{I}}$ then from definition, there is no conflict. Assume by contradiction that $t_{\vec{v}}^{\mathcal{I}} \neq t_{\vec{w}}^{\mathcal{I}}$ and for each $1 \le i \le n$, $f_i^{\mathcal{I}}(t_{\vec{v}}^{\mathcal{I}}) = f_i^{\mathcal{I}}(t_{\vec{w}}^{\mathcal{I}})$. Since $\mathcal{I} \models \sigma''(\mathcal{A})$, then for each $1 \le i \le n$, $v_i^{\mathcal{I}} = w_i^{\mathcal{I}}$. Hence $v_i^{\mathcal{I}} \in Q_{w_i}^{\mathcal{I}}$ which implies

$$\{t_{\vec{v}}^{\mathcal{I}}, t_{\vec{w}}^{\mathcal{I}}\} \subseteq (\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap \exists f_n.Q_{w_n})^{\mathcal{I}}.$$

Since $w_1^{\mathcal{I}}$ appears as the first component of two distinct reified tuples that satisfy $\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap \exists f_n.Q_{w_n}$, it yields $w_1^{\mathcal{I}} \notin (\le f_1^-.(\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap \exists f_n.Q_{w_n}))^{\mathcal{I}}$. This is contradiction to the assumption that $\mathcal{I} \models f''(\mathcal{A})$, because axiom $w_1 :\le f_1^-(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n}) \in f''(\mathcal{A})$.

□

Definition 5.3.2. *(disjoint union model property of $\mathcal{SHIQ}$ DL) Let C be an $\mathcal{SHIQ}$ concept and $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$, $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2})$ be two models of C. Then the interpretation $\mathcal{I}_1 \uplus \mathcal{I}_2 = (\Delta^{\mathcal{I}_1} \uplus \Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_1} \uplus \cdot^{\mathcal{I}_2})$ which is disjoint union of $\mathcal{I}_1$ and $\mathcal{I}_2$ is also a model of C.*

Lemma 5. *(Existence of tuple-admissible model) Let $\phi(\mathcal{K})$ be reified counterpart of some $\mathcal{NDL}$ knowledge base $\mathcal{K}$ according to ABox reification algorithm. If $\phi(\mathcal{K})$ is satisfiable, then it has tuple-admissible model.*

Proof. Let $\mathcal{I}$ be model $\phi(\mathcal{K}) = \langle \mathcal{S}, \mathcal{A} \rangle$. New tuple-admissible model $\hat{\mathcal{I}}$ of $\phi(\mathcal{K})$ will be constructed from $\mathcal{I}$ by fixing all conflicts of tuple representatives.

For $2 \leq n \leq n_{max}$ and n-tuple $\vec{d} = \langle d_1, ..., d_n \rangle \in (\top_1^{\mathcal{I}})^n$, the set of all reifications of tuple $\vec{d}$ can be defined as

$$T_{\vec{d}} = \{t \in \Delta^{\mathcal{I}} \mid \langle t, d_1 \rangle \in f_1^{\mathcal{I}} \wedge ... \wedge \langle t, d_n \rangle \in f_n^{\mathcal{I}}\}.$$

Each set $T_{\vec{d}}$, that contains more than one element, represents set of conflicting individuals for the tuple. For any such set, an arbitrary element $t_{\vec{d}} \in T_{\vec{d}}$ is chosen. Let $Conf$ denotes the set of all conflicting individuals except the chosen ones.

Before transforming $\mathcal{I}$ into an interpretation $\hat{\mathcal{I}}$ that contains no conflicts, consider simple case, where there is only single conflicting element in $Conf$ and no *ABox* individuals. This conflict can be resolved as follows.

Let $\mathcal{I}'$ be an interpretation consisting of two disjoint copies of $\mathcal{I}$. $\mathcal{I}'$ contains the conflicting element t and a copy t' of t. $\hat{\mathcal{I}}$ can be defined from $\mathcal{I}'$ by the operation on $\mathcal{I}'$, which change the interpretation of f_1 under $\mathcal{I}'$ as follows:

$$f_1^{\mathcal{I}'} = (f_1^{\mathcal{I}'} \setminus \{\langle t, f_1^{\mathcal{I}'}(t) \rangle, \langle t', f_1^{\mathcal{I}'}(t') \rangle\}) \cup \{\langle t, f_1^{\mathcal{I}'}(t') \rangle, \langle t', f_1^{\mathcal{I}'}(t) \rangle$$

It will be referred to this operation as *exchanging* $f_1^{\mathcal{I}'}(t)$ *with* $f_1^{\mathcal{I}'}(t')$. Note that this operation preserves the interpretation of all other atomic concepts and roles and that resulting interpretation contains no more conflicting elements.

The construction in general case is little bit more complicated, since one needs to consider *ABox* axioms and $Conf$ may be of arbitrary cardinality. However using the Lemma 5 , it is possible to avoid interference of *ABox* axioms. Since there are no two *ABox* individuals that are conflicting with each other, in each set $T_{\vec{d}}$ there is at most one element that appears as the image of an *ABox* individual of the interpretation $\mathcal{I}$. Hence, $Conf$ can be constructed without any elements that appear as images of *ABox* individuals of $\mathcal{I}$.

Let $\mathcal{I}'$ denote the disjoint union of $\natural(2^{Conf})$ copies of $\mathcal{I}$ and $d_{\mathcal{Z}}$ the copy of $d \in \Delta^{\mathcal{I}}$ in the $\mathcal{Z}$-th copy of $\mathcal{I}$, where the set $\mathcal{Z} \subseteq Conf$. $\hat{\mathcal{I}}$ is defined from $\mathcal{I}'$ as the result of *simultaneously* exchanging $f_1^{\mathcal{I}'}(d_{\mathcal{Z}})$ with $f_1^{\mathcal{I}'}(d_{\mathcal{Z} \setminus \{d\}})$, for each $d \in Conf$ and each $\mathcal{Z} \subseteq Conf$ with $d \in \mathcal{Z}$.

From the construction of $\hat{\mathcal{I}}$ it is easy to show that

1. $\hat{\mathcal{I}}$ does not contain any conflicts.

2. If C is a $\mathcal{SHIQ}$-concept, $d \in C^{\mathcal{I}}$ and $\mathcal{Z} \subseteq Conf$, then $d_{\mathcal{Z}} \in C^{\mathcal{I}}$.

For detailed proofs of above statements see [47]. From (2) it follows that $\hat{\mathcal{I}} \models \mathcal{S}$ and it remains to fix interpretation of the *ABox*. This can be done by interpreting all individuals in a single copy, e.g., by setting $w^{\hat{\mathcal{I}}} = w^{\mathcal{I}}_{\emptyset}$, for each ABox individual w.

For every assertion $w : C \in \mathcal{A}$ from (2) it follows that $w^{\mathcal{I}} \in C^{\mathcal{I}}$ implies $w^{\hat{\mathcal{I}}} \in C^{\hat{\mathcal{I}}}$. Since there are no ABox individuals in $Conf$, the interpretation for f_i is not changed for $w^{\hat{\mathcal{I}}}_{\emptyset}$. Hence, $\langle w_1^{\hat{\mathcal{I}}}, w_2^{\hat{\mathcal{I}}} \rangle \in f_i^{\hat{\mathcal{I}}}$ holds for every assertion $\langle w_1, w_2 \rangle \in f_i \in \mathcal{A}$. Thus $\hat{\mathcal{I}} \models \mathcal{S}$ and finally $\hat{\mathcal{I}} \models \phi(\mathcal{K})$.

□

Lemma 6. *(Completeness of encoding) Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base and $\phi(\mathcal{K})$ its reified counterpart according to the ABox reification algorithm. If $\phi(\mathcal{K})$ is satisfiable, then $\mathcal{K}$ is satisfiable.*

Proof. From Lemma 5, let $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ be *tuple-admissible model.* An interpretation $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$ can be build from $\mathcal{I}'$ as follows:

1. $\Delta^{\mathcal{I}} = \top_1^{\mathcal{I}'}$;

2. $\mathbf{P}^{\mathcal{I}} = \{(d_1, ..., d_n) \mid \exists t \in A_{\mathbf{P}}^{\mathcal{I}'}.((t, d_i) \in f_i^{\mathcal{I}'}, \text{for } i \in \{1, ..., n\})\}$, for each relation representative $A_{\mathbf{P}}$ included in $\top_n$;

3. $A^{\mathcal{I}} = A^{\mathcal{I}'}$, for each atomic concept A included in $\top_1$;

4. $w^{\mathcal{I}} = w^{\mathcal{I}'}$, for each individual w from $\top_1$

It is easy to verify that constructed interpretation $\mathcal{I}$ is model of $\mathcal{K}$. □

Theorem 2. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base and $\phi(\mathcal{K})$ its reified counterpart according to ABox reification algorithm. Then $\mathcal{K}$ is satisfiable, if and only if $\phi(\mathcal{K})$ is satisfiable.*

Proof. The proof is immediate consequence of Lemma 3 and Lemma 6.

□

5.3.2 Reduction to $\mathcal{SHIF}$ DL

The following modification of the reification algorithm solves tuple-admissibility problem by creating new role and adding one general concept inclusion axiom for each tuple. The only *cardinality restrictions* used in new axioms are *functional unqualified cardinality restrictions*. Therefore, the reification algorithm outputs binary KB of expressivity $\mathcal{SHIF}$.

Construction 5.3.2. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base, $\mathcal{W}_{\mathcal{A}}$ is set of all tuples occurring in $\mathcal{A}$. The functional reified counterpart $\varphi(\mathcal{K})$ of $\mathcal{K}$ is constructed as follows:*

$$f''_{2\varphi}(\mathcal{A}) = \{g_{\vec{w}} \sqsubseteq f_1 \mid \vec{w} \in \mathcal{W}_{\mathcal{A}}\} \cup \tag{5.11}$$
$$\{\top \sqsubseteq \leq 1 g^{-}_{\vec{w}} \mid \vec{w} \in \mathcal{W}_{\mathcal{A}}\} \cup \tag{5.12}$$
$$\{\top_n \sqcap \exists f_2.Q_{w_2} \sqcap \ldots \sqcap f_n.Q_{w_n} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1} \mid \vec{w} = \langle w_1, \ldots, w_n \rangle \in \mathcal{W}_{\mathcal{A}}\} \tag{5.13}$$

The set of axioms $f''_{2\varphi}(\mathcal{A})$ ensures tuple-admissibility of asserted tuples, $g_{\vec{w}}$ is newly introduced role for each tuple $\vec{w}$. Finally, the functional reified counterpart $\varphi(\mathcal{K})$ is defined as follows:

$$\varphi(\mathcal{K}) \quad := \quad \langle \sigma(\mathcal{S}) \cup f(\mathcal{S}) \cup \sigma''(\mathcal{A}) \cup f''_{2\varphi}(\mathcal{A}), f''_1(\mathcal{A}) \rangle$$

□

Lemma 7. *(Tuple-admissibility of asserted individuals) Let $\varphi(\mathcal{K})$ be reified counterpart of some $\mathcal{NDL}$ knowledge base $\mathcal{K}$ according to ABox reification algorithm. If $\varphi(\mathcal{K})$ satisfiable, then there is no conflict between any two tuple representatives of the ABox.*

Proof. Let $\mathcal{I}$ be model of $\varphi(\mathcal{K})$ and $t^{\mathcal{I}}_{\vec{w}}$, $t^{\mathcal{I}}_{\vec{v}}$ two individuals of ABox. Assume by contradiction that $t^{\mathcal{I}}_{\vec{v}} \neq t^{\mathcal{I}}_{\vec{w}}$ and for each $1 \leq i \leq n$, $f^{\mathcal{I}}_i(t^{\mathcal{I}}_{\vec{v}}) = f^{\mathcal{I}}_i(t^{\mathcal{I}}_{\vec{w}})$. Since $\mathcal{I} \models \sigma''_{\varphi}(\mathcal{A})$, then for each $1 \leq i \leq n$, $v^{\mathcal{I}}_i = w^{\mathcal{I}}_i$. Hence $v^{\mathcal{I}}_i \in Q^{\mathcal{I}}_{w_i}$ which implies

$$\{t^{\mathcal{I}}_{\vec{v}}, t^{\mathcal{I}}_{\vec{w}}\} \subseteq (\top_n \sqcap \exists f_2.Q_{w_2} \sqcap \ldots \sqcap \exists f_n.Q_{w_n})^{\mathcal{I}}.$$

and thus $\{t^{\mathcal{I}}_{\vec{v}}, t^{\mathcal{I}}_{\vec{w}}\} \subseteq \exists g_{\vec{w}}.Q_{w_1}$ (5.13). Let a_1 and a_2 be elements of Q_{w_1} that are connected through $g_{\vec{w}}$ to $t^{\mathcal{I}}_{\vec{v}}$ and $t^{\mathcal{I}}_{\vec{w}}$ respectively. Since f is functional and $g^{\mathcal{I}} \subseteq f^{\mathcal{I}}$, it follows (5.11) that $a_1 = a_2 = v^{\mathcal{I}}_1$. Hence, $g^{-}_{\vec{w}}$ connects two distinct elements to $v^{\mathcal{I}}_1$, which is contradiction to the assumption that $\mathcal{I} \models f''_{\varphi}(\mathcal{A})$, because of the axiom (5.12). □

Theorem 3. *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base and $\varphi(\mathcal{K})$ its reified counterpart according to ABox reification algorithm. Then $\mathcal{K}$ is satisfiable, if and only if $\varphi(\mathcal{K})$ is satisfiable.*

Proof. For the only-if direction it can be used construction from Lemma 3. In addition to the construction, we must define interpretation of newly introduced roles $g_{\vec{w}}$ such that $g_{\vec{w}}^{\mathcal{I}'} = \{(t_{(d_1,...,d_n)}, d_1) \mid (d_1, ..., d_n) \in T_n^{\mathcal{I}}\}$ for each tuple $(d_1, ..., d_n) = w^{\mathcal{I}}$ of element domains. Then, it is easy to show that this interpretation is model of $\varphi(\mathcal{K})$.

For the converse direction, we must prove existence of tuple-admissible model of $\varphi(\mathcal{K})$ and then use construction from Lemma 6 to create model of $\mathcal{K}$. Existence of tuple-admissible model of $\varphi(\mathcal{K})$ is proved by Lemma 5 if we use Lemma 7 to address tuple-admissibility of asserted individuals.

□

5.4 Complications and contribution

As explained at the beginning of this chapter choices for expressivity of reified counterpart were limited by supported expressivity of state-of-art reasoners. This section discusses different modifications of reification algorithms that were not sufficient for our purpose and summarizes contribution of the chapter.

5.4.1 Nominals

In the reification algorithm using *TBox* (Section 5.2), representative concepts A_w and $T_{\vec{w}}$ were used to describe properties of $\mathcal{NDL}$ individuals and tuples respectively. Complex framework was created to single out representative concepts in order to get tuple-admissible model. Intuitively, it would be natural to introduce new individuals $t_{\vec{w}}$ for each tuple and instead of the complex framework use axioms:

$A_w \equiv \{w\}$, for each individual w occurring in reified counterpart

$T_{\vec{w}} \equiv \{t_{\vec{w}}\}$, for each tuple $t_{\vec{w}}$ occurring in reified counterpart

From the above axioms it is obvious that all required properties of representative concepts are satisfied, because each of the representative concepts is interpreted by exactly one individual.

However, proof of the reification algorithm is based on *tree-model property* of underlying binary DL. In [29] it was shown that DL with *nominals*, *number restrictions*, and *inverse roles* looses *tree-model property*[3]. Note that in reification of schema we used both *number restrictions* (5.4) and *inverse roles* (5.1). Thus, proof of the reification algorithm using *TBox* cannot be used with presence of *nominals* in reified counterpart.

5.4.2 Reduction of cardinality restrictions

In the reification algorithm using *ABox* we were able to reduce the expressivity of reified counterpart from $\mathcal{SHIQ}$ to $\mathcal{SHIF}$. The reification algorithm introduced new role $g_{\vec{w}}$ and added one general concept inclusion axiom for each tuple $\vec{w}$. Each of new roles was defined as restricted subrole of role f_1 by axiom $g_{\vec{w}} \sqsubseteq f_1$.

This technique, however, cannot be used for the reification using *TBox*. The problem is that correctness of the extension of Fischer-Ladner closure (Definition 5.2.3) is based on $\mathcal{CIQ}$ DL. However, $\mathcal{CIQ}$ DL does not allow role inclusion axioms (such as $g_{\vec{w}} \sqsubseteq f_1$). Note that there are some role inclusion axioms but only in the definition of universal accessibility role (Definition 5.2.1). Therefore these axioms are not included in set of axioms from which extended Fischer-Ladner closure is evaluated.

5.4.3 Summary

The reification algorithm according to Calvanese at al. was firstly published in [37] as a part of the framework for solving QCP. The output binary logics of the reification algorithm was Converse Propositional Dynamic Logics (CPDLg). Later it was found out direct correspondence between logics CPDLg and $\mathcal{CIQ}$ DL. Calvanese at. al published short paper about QCP in context of $\mathcal{CIQ}$ DL in [38].

The Section 5.2 describes this reification algorithm in context of DL in greater detail. In particular, the algorithm outputs knowledge base in $\mathcal{SHIQ}$ DL. Due to different expressivity of reified counterparts we were not able to define so-called universal accessibility role as defined in Calvanese work. However we successfully used pseudo-universal accessibility role (see Definition 5.2.1) that is superset of universal accessibility role. Correctness of pseudo-universal accessibility role is explained in Lemma 2.

In the Section 5.3 we introduced two different reification algorithms. The first algorithm is only syntactical modification of the one defined by Horrocks at al. [47]. The second defines the reduction of the algorithm to less

[3]*tree-model property* holds only in DLs with nominals that excludes one of *number restrictions* or *inverse roles* [31, 32, 33]

expressive description logics i.e. $\mathcal{SHIF}$ DL. As $\mathcal{SHIF}$ DL is also a subset of underlying logics of OWL-DL, this reification algorithm has better support in various ontology tools than original one. Definition of the new algorithm can be found in Construction 5.3.2 and formal proof is done in Theorem 3.

6 Acquisition of benchmark data

For comparison of reification algorithms discussed in Chapter 5 benchmark $\mathcal{NDL}$ knowledge bases are needed. Manual construction of such ontologies would be very irritating and time-consuming task. Another way to get around with it is to construct benchmark ontologies from existing ones. Next section of this chapter presents automatic transformation algorithm for OWL-DL ontologies. Last section construct extension to this algorithm towards n-ary relations by semi-automatic algorithm.

6.1 Converting OWL-DL to NDL

As explained in previous chapters, OWL-DL uses properties to express relationships between classes. Although properties are only binary relations over classes, W3C proposed solution for defining n-ary relations in [6]. In the document, W3C recognized 3 different types of situations (cases), where n-ary relation patterns are applicable. For each case, it provides a way to solve the problem using reification of n-ary relation, thus representing relations by classes.

Since $\mathcal{NDL}$ can express n-ary relations directly via relation constructors, conversion from OWL-DL ontologies to $\mathcal{NDL}$ should also be able to join multiple OWL-DL properties to form one n-ary $\mathcal{NDL}$ relation. Some basic requirements of OWL-DL conversion are demonstrated in Figure 6.1. Left side of the diagram shows example of n-ary relation pattern of the W3C document for defining n-ary relations. In the example, it is expressed that "Steve has temperature, which is elevated, but falling". `Steve` is instance (individual) of class *Person*, `elevated` and `falling` are instances of classes *TemperatureValue* and *TemperatureTrend*. The knowledge could have been modelled by two properties, **hasTemperatureLevel** and **hasTemperatureTrend**, both relating to *Person*, however W3C recommends to relate `Steve` to the complex object instead. The complex object, `temperatureObservation1`, is instance of class *TemperatureObservation* and represents different facts about `Steve`'s temperature. Reason for this modelling choice lies in the fact that in intended interpretation of properties, temperature level and temperature trend are inextricably intertwined. The class *TemperatureObservation* serves as reifying class of ternary relation. Right side of the figure shows converted knowledge about Steve in $\mathcal{NDL}$. Names of OWL-DL classes *Person*, *TemperatureValue*, *TemperatureTrend* and individuals `Steve`, `elevated`, `falling` are converted to same names in $\mathcal{NDL}$. OWL-DL properties **hasTemperature**, **temperatureValue**, **temperatureTrend** are chained together through individuals of concept *TemperatureObservation* to form new $\mathcal{NDL}$ relation **Temperature** of arity 3. First argument of the relation corresponds to domain of property **hasTemperature**, second and third arguments corresponds to ranges of properties **temperatureValue** and **temperatureTrend**.

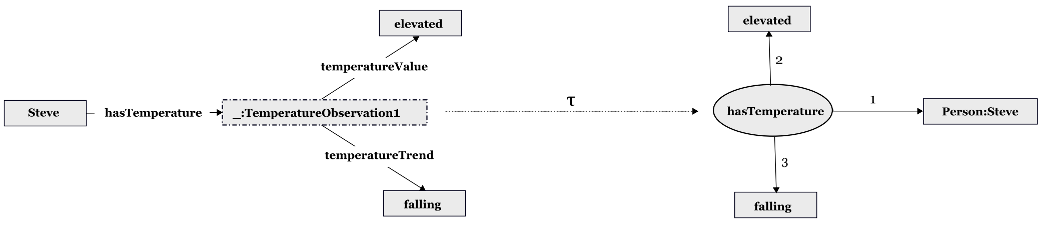

Figure 6.1: Example of transformation from OWL-DL reifying class to $\mathcal{NDL}$ relation

Because of restricted power of $\mathcal{NDL}$ some semantics behind OWL-DL axioms can be captured only partially, some cannot be captured at all. In both OWL-DL and $\mathcal{NDL}$ it is not assumed that individuals with different names are mapped to different elements in domain[1]. However, contrary to $\mathcal{NDL}$, OWL-DL is able to express equalities and inequalities among individuals. In the transformation, we will define new representative concept B_o for each individual o and state $o : B_o$. Equalities among individuals will be modelled in $\mathcal{NDL}$ by equivalence of its representative concepts. On the other hand, for inequalities, disjointness of representatives will be used. Moreover, OWL-DL allows to refer to individuals in class constructors (enumerated class, property value restriction). Classes that refer to individuals will be converted to concepts that refer to the representatives of individuals.

Another source of problems for the conversion comes from the inability of $\mathcal{NDL}$ to define cardinalities. In the transformation, we will define new representative concept $B_{\leqslant n\mathbf{r}}$ for each minimal cardinality restriction $\leqslant n\mathbf{r}$ occurring in OWL-DL ontology. For each representative concept some additional axioms will be added to $\mathcal{NDL}$ knowledge base to preserve properties

[1]This is called unique name assumption (UNA)

of minimal cardinality restrictions as much as possible. Moreover, using representatives of minimal cardinality restrictions, it will possible to define cardinalities of type $\geqslant n\mathbf{r}$ and $= n\mathbf{r}$.

In OWL-DL, property restriction class constructor allows to define a new class from domain of an object property restricted over its range. Also definition of a new class from range of an object property is possible by applying property restriction on inverse of the object property. Although $\mathcal{NDL}$ cannot define inverse of binary relations, it can return and restrict any argument of relation by projection constructor and restriction constructor respectively. For this reason, each construct of OWL-DL that uses inverse property, can be transformed to equivalent construct in $\mathcal{NDL}$. Example of such transformations are in Figure 6.2. In the figure an OWL-DL property **isChildOf** is defined as an inverse property of **hasChild**. *Parent* is defined as class that has at least one value in the **hasChild** property. *RichChild* is defined as class that has at least one value in property of **isChildOf** with the type *RichPerson*. $\mathcal{NDL}$ translate both properties **hasChild** and **isChildOf** to only one relation **ParentChildRelation**.

$$\begin{array}{rcl}
\mathbf{hasChild} & - & \mathit{property} \\
\mathbf{isChildOf} & \equiv & \mathbf{hasChild}^{-} \\
\mathit{Parent} & \sqsubseteq & \exists\mathbf{hasChild}.\top \\
\mathit{RichChild} & \sqsubseteq & \exists\mathbf{isChildOf}.\mathit{RichPerson}
\end{array}$$

$$\begin{array}{rcl}
\mathbf{Parentship} & - & \mathit{relation} \\
\mathit{Parent} & \sqsubseteq & \exists[\$1]\mathbf{Parentship} \\
\mathit{RichChild} & \sqsubseteq & \exists[\$2](\mathbf{ParentShip} \sqcap (1/2) : \mathit{RichPerson})
\end{array}$$

Figure 6.2: Equivalence of OWL-DL and $\mathcal{NDL}$ constructs

As showed in the example, in the transformation from OWL-DL to $\mathcal{NDL}$, property and inverse of property has to be defined as one $\mathcal{NDL}$ relation. To restrict transformation parameter to only correct definitions in this sense, a well formed transformation parameter will be introduced.

Whole transformation of OWL-DL ontology to $\mathcal{NDL}$ knowledge base will be recursively defined by a function τ. Transformation function τ depends on a triple $\langle \nu, \delta, \rho \rangle$ called transformation parameter. In the triple, ν is function for conversion of OWL-DL atomic entities to $\mathcal{NDL}$. The function converts each atomic class to $\mathcal{NDL}$ concept, atomic property to

$\mathcal{NDL}$ relation and instance to $\mathcal{NDL}$ individual. The correspondence of domain and range of OWL-DL properties in $\mathcal{NDL}$ relations will be expressed by functions δ and ρ. Both functions take as an argument an OWL-DL property and return a positive integer. If $\mathbf{r}$ (possibly inverse property) is an OWL-DL property that is mapped to an n-ary relation $\mathbf{R}$, then δ returns index of argument in $\mathbf{R}$ that represents the domain of $\mathbf{r}$ and ρ returns index of argument in $\mathbf{R}$ that represents the range of $\mathbf{r}$. In the example from figure Figure 6.1, $\delta(\mathbf{hasTemperature}) = 1, \rho(\mathbf{temperatureValue}) = 2, \rho(\mathbf{temperatureTrend}) = 3$.

As explained in Chapter 2, OWL-DL without datatypes support can be transformed to $\mathcal{SHOIN}$ DL. Exact transformation of OWL-DL axioms and facts to $\mathcal{SHOIN}$ DL axioms is given in Table 2.3. Thanks to this correspondence, the transformation from OWL-DL ontology to $\mathcal{NDL}$ knowledge base can be described in two steps. First, all OWL-DL axioms and facts are converted to appropriate $\mathcal{SHOIN}$ DL axioms. Second, $\mathcal{SHOIN}$ DL knowledge base is converted to $\mathcal{NDL}$ knowledge base.

In the following text, instead of describing transformation from an OWL-DL ontology to an $\mathcal{NDL}$ knowledge base, transformation from a $\mathcal{SHOIN}$ DL knowledge base to $\mathcal{NDL}$ knowledge base will be used. Every concept, role and individual of $\mathcal{SHOIN}$ DL knowledge base directly corresponds to a class, a property and an instance of the OWL-DL ontology respectively.

Transformation from $\mathcal{SHOIN}$ DL to $\mathcal{NDL}$ will be parametrized by so called transformation parameter (TP). TP will be used to define a particular mapping between concepts of both KBs. To ensure that the transformation produces correct $\mathcal{NDL}$ KB, the notation of a well-formed TP needs to be defined.

For simplification of the rules in the transformation, extended transformation parameter $\langle \nu, \delta, \rho \rangle$ will be used, where each of the functions ν, δ, ρ are extensions of the above defined ones by inverses of atomic roles.

Definition 6.1.1. *[transformation parameter] Given $\mathcal{K}$ a $\mathcal{SHOIN}$ DL knowledge base, a transformation parameter of $\mathcal{K}$ is a triple $\mathcal{M} = \langle \nu, \delta, \rho \rangle$, such that ν is a function that map each atomic concept of $\mathcal{K}$ to atomic $\mathcal{NDL}$ concept, each atomic role of $\mathcal{K}$ to atomic $\mathcal{NDL}$ relation and each individual of $\mathcal{K}$ to $\mathcal{NDL}$ individual, δ (domain-index function) and ρ (range-index function) are functions that maps each atomic role of $\mathcal{K}$ to a positive integer.*

Naturally extended transformation parameter $\mathcal{M}' = \langle \nu', \delta', \rho' \rangle$ of $\mathcal{K}$ is given by

$$\begin{array}{ccc} \nu'(\mathbf{p}) = \nu(\mathbf{p}) & \delta'(\mathbf{p}) = \delta(\mathbf{p}) & \rho'(\mathbf{p}) = \rho(\mathbf{p}) \\ \nu'(\mathbf{p}^-) = \nu(\mathbf{p}) & \delta'(\mathbf{p}^-) = \rho(\mathbf{p}) & \rho'(\mathbf{p}^-) = \delta(\mathbf{p}) \end{array}$$

Additionally we say that a transformation parameter $\mathcal{M}$ is well formed if $1 \leq \delta'(\mathbf{r}) \leq m$ for each expression $(\leq \mathbf{r})$, $(\geq \mathbf{r})$, or $(= \mathbf{r})$ occurring in $\mathcal{K}$ and $1 \leq \delta'(\mathbf{r}), \rho'(\mathbf{r}) \leq m$ for each expression $(\exists \mathbf{r}.C)$ or $(\forall \mathbf{r}.C)$ occurring in $\mathcal{K}$, where $\mathbf{r}$ is an arbitrary role with arity m, C is arbitrary concept and n is arbitrary non-negative integer.

Note that TP is well-formed if its natural extension satisfy certain conditions on domain-index and range-index function. In the following text we define the transformation function formally.

Definition 6.1.2 (transformation function of concept, role, individual, schema, *ABox*)**.** *Given $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ a $\mathcal{SHOIN}$ DL knowledge base with naturally extended well-formed transformation parameter $\mathcal{M} = \langle \nu, \delta, \rho \rangle$, A (atomic), C concepts and $\mathbf{p}$ (atomic), $\mathbf{r}$ roles with arity m, a transformation function $\tau_{\mathcal{M}}$ of concepts, properties and individuals in $\mathcal{K}$ is given by*

$$\begin{aligned} \tau_{\mathcal{M}}(A) &= \nu(A) & \tau_{\mathcal{M}}(\neg C) &= \neg \tau_{\mathcal{M}}(C) \\ \tau_{\mathcal{M}}(\mathbf{p}) &= \nu(\mathbf{p}) & \tau_{\mathcal{M}}(\leqslant n\mathbf{r}) &= B_{\leqslant n\mathbf{r}} \\ \tau_{\mathcal{M}}(\mathbf{p}^-) &= \nu(\mathbf{p}) & \tau_{\mathcal{M}}(\geqslant n\mathbf{r}) &= \neg B_{\leqslant (n-1)\mathbf{r}} \\ \tau_{\mathcal{M}}(o) &= \nu(o) & \tau_{\mathcal{M}}(\{o_1, ..., o_n\}) &= B_{o_1} \sqcup ... \sqcup B_{o_n} \end{aligned}$$

$$\begin{aligned} \tau_{\mathcal{M}}(\exists \mathbf{r}.C) &= \exists[\delta^{\mathbf{r}}](\tau_{\mathcal{M}}(\mathbf{r}) \sqcap (\rho^{\mathbf{r}}/m : \tau_{\mathcal{M}}(C))) \\ \tau_{\mathcal{M}}(\forall \mathbf{r}.C) &= \forall[\delta^{\mathbf{r}}](\tau_{\mathcal{M}}(\mathbf{r}) \sqcap (\rho^{\mathbf{r}}/m : \tau_{\mathcal{M}}(C))) \\ \tau_{\mathcal{M}}(C_1 \sqcap ... \sqcap C_n) &= \tau_{\mathcal{M}}(C_1) \sqcap ... \sqcap \tau_{\mathcal{M}}(C_n) \\ \tau_{\mathcal{M}}(C_1 \sqcup ... \sqcap C_n) &= \tau_{\mathcal{M}}(C_1) \sqcup ... \sqcup \tau_{\mathcal{M}}(C_n) \end{aligned}$$

$B_{\leqslant i\mathbf{r}}$, B_{o_i} are newly introduced $\mathcal{NDL}$ concepts.

Note that since $\mathcal{NDL}$ can express restrictions on any argument of a relation, $\tau_{\mathcal{M}}(\mathbf{p})$ and $\tau_{\mathcal{M}}(\mathbf{p}^-)$ may both convert to the same $\mathcal{NDL}$ relation $\nu(\mathbf{p})$ while specifying range of a role is left on range-index function δ. In the transformation, domain of role $\mathbf{r}$ corresponds to $\delta^{\mathbf{r}}$-th argument of $\mathcal{NDL}$ relation $\tau_{\mathcal{M}}(\mathbf{r})$, range of role $\mathbf{r}$ corresponds to $\rho^{\mathbf{r}}$-th argument of $\mathcal{NDL}$ relation $\tau_{\mathcal{M}}(\mathbf{r})$.

For use of nominals and for expressing of equalities and inequalities of individuals in *ABox* axioms, new concepts B_{o_i} are introduced. Note that for expressing inequalities of individuals o_1 and o_2, axiom $B_{o_1} \sqsubseteq \neg B_{o_2}$ is sufficient, because neither B_{o_1} nor B_{o_2} can become unsatisfiable (in transformation of whole $\mathcal{SHOIN}$ DL knowledge base, axioms of type $o : B_o$ will be added for each individual representative B_o).

Before definition of the transformation function over the whole $\mathcal{SHOIN}$ DL knowledge base, some helper functions need to be defined. Function *one* ensures desired properties of individual representative concepts in the transformation. On the other hand, function *card* is used to express additional properties of cardinality representative concepts.

Definition 6.1.3 (functions *one* and *card*)**.** *Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{SHOIN}$ DL knowledge base with transformation parameter $\mathcal{M}$. The functions $one(\mathcal{K}, \mathcal{M})$, $card(\mathcal{K})$ are given by*

$$\begin{aligned}
one(\mathcal{K},\mathcal{M}) =& \{o : B_o \mid B_o \in \mathcal{X}\} \\
card(\mathcal{K},\mathcal{M}) =& \{B_{\leqslant 0\mathbf{r}} \equiv \forall[\delta^{\mathbf{r}}](\tau_{\mathcal{M}}(\mathbf{r}) \sqcap (\rho^{\mathbf{r}}/m : \bot)) \mid \mathbf{r} \textit{ with arity } m \textit{ occurs in } \mathcal{K}\} \\
& \cup \{B_{\leqslant i\mathbf{r}} \sqsubseteq B_{\leqslant j\mathbf{r}} \mid B_{\leqslant i\mathbf{r}}, B_{\leqslant j\mathbf{r}} \in \mathcal{Y} \wedge i < j \wedge \neg\exists j' : \\
& \quad (i < j' < j \wedge B_{\leqslant j'\mathbf{r}} \in \mathcal{Y})\}
\end{aligned}$$

where

$$\begin{aligned}
\mathcal{X} =& \{B_o \mid B_o \textit{ is individual representative occurring in } \tau_{\mathcal{M}}(\mathcal{S}) \cup \tau_{\mathcal{M}}(\mathcal{A})\} \\
\mathcal{Y} =& \{B_{\leqslant i\mathbf{r}} \mid B_{\leqslant i\mathbf{r}} \textit{ is cardinality representative occurring in } \tau_{\mathcal{M}}(\mathcal{S}) \cup \tau_{\mathcal{M}}(\mathcal{A})\} \\
& \cup \{B_{\leqslant 0\mathbf{r}} \mid \mathbf{r} \textit{ occurs in } \mathcal{K}\}
\end{aligned}$$

Intuitively, from $\mathcal{SHOIN}$ DL cardinality restrictions can be deduced that $(\leq 0\mathbf{r}) \sqsubseteq (\leq 1\mathbf{r}) \sqsubseteq (\leq 2\mathbf{r}) \sqsubseteq ...$, $card(\mathcal{K})$ defines same hierarchy among cardinality representatives $B_{\leq i\mathbf{r}}$ of each relation $\mathbf{r}$. In addition it defines lowest elements in these hierarchies ($B_{\leq 0\mathbf{r}}$), according to the fact that in $\mathcal{SHOIN}$ DL $(\leq 0\mathbf{r}) \equiv \forall\mathbf{r}.\bot$.

For automatic transformation from $\mathcal{SHOIN}$ DL to $\mathcal{NDL}$, transformation parameter (TP) of $\mathcal{SHOIN}$ DL has to be created automatically. On the other hand, for semi-automatic transformation, it might be useful for an user to be able to define some parts of TP and let automatic procedure to complete the rest. Such an algorithm for defining well-formed TP is showed in Algorithm 1. Input of the algorithm is $\mathcal{SHOIN}$ DL knowledge base $\mathcal{K}$ with transformation parameter $\langle \nu, \delta, \rho \rangle$ and some linear order $\leq$ on all atomic roles of $\mathbf{K}$. The transformation parameter can be defined partially, however for each role $\mathbf{p}$ of $\mathcal{SHOIN}$ DL knowledge base, either all or none of ν, δ, ρ has to be specified. Output of the algorithm is TP that is defined on all atomic concepts, roles and individuals. In the figure, $min(\mathcal{W})$ is function that returns minimal element in $\mathcal{W}$ according to $\leq$, $pop(\mathcal{Q}_i)$ returns and remove one element from $\mathcal{Q}_i$.

6.1.1 Automatic transformation

In automatic transformation, at first Algorithm 1 will be ran on $\mathcal{SHOIN}$ DL knowledge base $\mathcal{K}$ with empty transformation parameter and arbitrary ordering $\leq$. Product of this algorithm is a well-formed TP that will be naturally extended according to Definition 6.1.1 and used as an input parameter for transformation function $\tau'_{\mathcal{M}}$ defined below.

Definition 6.1.4 (binary transformation function of schema, *ABox*, knowledge base)**.** *Given $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ a $\mathcal{SHOIN}$ DL knowledge base with naturally extended well-formed transformation parameter $\mathcal{M} = \langle \nu, \delta, \rho \rangle$, C, C_1, C_2 concepts and $\mathbf{r}$, $\mathbf{r_1}$, $\mathbf{r_2}$ roles of arity m, a binary transformation function of $\mathcal{S}$ and $\mathcal{A}$ is given by*

$$
\begin{aligned}
\tau'_{\mathcal{M}}(\mathcal{S}) =& \{\tau_{\mathcal{M}}(C_1) \sqsubseteq \tau_{\mathcal{M}}(C_2) \mid (C_1 \sqsubseteq C_2) \in \mathcal{S}\} \\
& \cup \{\tau_{\mathcal{M}}(\mathbf{r_1}) \sqsubseteq \tau_{\mathcal{M}}(\mathbf{r_2}) \mid (\mathbf{r_1} \sqsubseteq \mathbf{r_2}) \in \mathcal{S} \wedge \langle \delta^{\mathbf{r_1}}, \rho^{\mathbf{r_1}} \rangle = \langle \delta^{\mathbf{r_2}}, \rho^{\mathbf{r_2}} \rangle\} \\
\tau'_{\mathcal{M}}(\mathcal{A}) =& \{\nu(o) : C \mid (o : C) \in \mathcal{A}\} \\
& \cup \{\langle \nu(o_1), \nu(o_2) \rangle : \tau_{\mathcal{M}}(\mathbf{r}) \mid (\langle o_1, o_2 \rangle : \mathbf{r}) \in \mathcal{A}\} \\
& \cup \{B_{o_1} \equiv B_{o_2} \mid (o_1 = o_2) \in \mathcal{A}\} \\
& \cup \{B_{o_1} \sqsubseteq \neg B_{o_2} \mid (o_1 \neq o_2) \in \mathcal{A}\}
\end{aligned}
$$

Binary transformation function of $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ is given by

$$
\tau'_{\mathcal{M}}(\mathcal{K}) = \langle \tau'_{\mathcal{M}}(\mathcal{S}) \cup \mathit{card}(\mathcal{K}, \mathcal{M}), \tau'_{\mathcal{M}}(\mathcal{A}) \cup \mathit{one}(\mathcal{K}, \mathcal{M}) \rangle
$$

The transformation straightforwardly convert each $\mathcal{SHOIN}$ DL axiom to $\mathcal{NDL}$ axiom. Additional axioms $\mathit{one}(\mathcal{K}, \mathcal{M})$ and $\mathit{card}(\mathcal{K}, \mathcal{M})$ ensure that representatives of nominals and cardinality restrictions partially satisfy their intention.

6.1.2 Semi-automatic transformation

Semi-automatic transformation allows to transform binary $\mathcal{SHOIN}$ DL knowledge base to n-ary $\mathcal{NDL}$ KB. The input of the transformation is definition of mappings from binary roles to n-ary roles. For this purpose, the notion of relation mapping is used.

The transformation can be constructed similarly as in the automatic transformation. Relation mappings define a transformation parameter partially. First, we construct arbitrary ordering $\leq$ on roles of the $\mathcal{SHOIN}$ DL KB where each role that is already defined in the transformation parameter is before any role that is not defined in TP. Second, such an ordering and transformation parameter is used as input for Algorithm 1. Third, product of this algorithm is a well-formed TP that will be naturally extended according to Definition 6.1.1 and used as an input parameter for transformation function $\tau''_{\mathcal{M}}$ defined below.

Definition 6.1.5 (n-ary transformation function of schema, *ABox*, knowledge base)**.** *Given $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ a $\mathcal{SHOIN}$ DL knowledge base with naturally extended well-formed transformation parameter $\mathcal{M} = \langle \nu, \delta, \rho \rangle$, C, C_1, C_2 concepts and $\mathbf{r}$, $\mathbf{r_1}$, $\mathbf{r_2}$ roles of arities $\epsilon^{\mathbf{r}}$, $\epsilon^{\mathbf{r_1}}$, and $\epsilon^{\mathbf{r_2}}$ respectively. An n-ary transformation function of $\mathcal{S}$ and $\mathcal{A}$ is given by*

$$\begin{aligned}
\tau''_{\mathcal{M}}(\mathcal{S}) =& \{\tau_{\mathcal{M}}(C_1) \sqsubseteq \tau_{\mathcal{M}}(C_2) \mid (C_1 \sqsubseteq C_2) \in \mathcal{S}\} \\
& \cup \{\tau_{\mathcal{M}}(\mathbf{r_1}) \sqsubseteq \tau_{\mathcal{M}}(\mathbf{r_2}) \mid (\mathbf{r_1} \sqsubseteq \mathbf{r_2}) \in \mathcal{S} \wedge \langle \delta^{\mathbf{r_1}}, \rho^{\mathbf{r_1}}, \epsilon^{\mathbf{r_1}} \rangle = \langle \delta^{\mathbf{r_2}}, \rho^{\mathbf{r_2}}, \epsilon^{\mathbf{r_2}} \rangle\} \\
\tau''_{\mathcal{M}}(\mathcal{A}) =& \{\nu(o) : C \mid (o : C) \in \mathcal{A}\} \\
& \cup \{\langle \nu(o_1), ..., \nu(o_n) \rangle : \tau_{\mathcal{M}}(\mathbf{r}) \mid \langle o_1, ..., o_n \rangle : \mathcal{T}(\mathbf{r})\} \\
& \cup \{B_{o_1} \equiv B_{o_2} \mid (o_1 = o_2) \in A\} \\
& \cup \{B_{o_1} \sqsubseteq \neg B_{o_2} \mid (o_1 \neq o_2) \in A\}
\end{aligned}$$

where $\mathcal{T}(\mathbf{r})$ is set of tuples that relate to $\tau_{\mathcal{M}}(\mathbf{r})$.

N-ary transformation function of $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ is given by

$$\tau''_{\mathcal{M}}(\mathcal{K}) = \langle \tau''_{\mathcal{M}}(\mathcal{S}) \cup card(\mathcal{K}, \mathcal{M}), \tau''_{\mathcal{M}}(\mathcal{A}) \cup one(\mathcal{K}, \mathcal{M}) \rangle$$

Note that the only relation inclusions for which $\langle \delta^{\mathbf{r_1}}, \rho^{\mathbf{r_1}}, \epsilon^{\mathbf{r_1}} \rangle = \langle \delta^{\mathbf{r_2}}, \rho^{\mathbf{r_2}}, \epsilon^{\mathbf{r_2}} \rangle$ holds are included in new n-ary knowledge base.

Finally, we defined notion of relation mapping that specifies only meaningful mappings from binary roles to n-ary relation.

Definition 6.1.6. *Relation mapping $\mathcal{M}$ is a tuple $\langle \mathbf{R}, \mathcal{W}, \delta, \rho \rangle$, where $\mathbf{R}$ is $\mathcal{NDL}$ relation with arity n_{max}, $\mathcal{W}$ is set of DL roles, δ is domain-index function and ρ is range-index function, such as*

- $|\mathcal{W}| \geq n_{max} - 1$
- $\forall \mathbf{p} \in \mathcal{W} : 1 \leq \delta(\mathbf{p}) \neq \rho(\mathbf{p}) \leq |\mathcal{W}| + 1$
- *If $\mathcal{G} = (V, E)$ is undirected graph, where $V = \{1, ..., |\mathcal{W}| + 1\}$ and $E = \{\{\delta(\mathbf{r}), \rho(\mathbf{r})\} \mid \mathbf{r} \in \mathcal{W}\}$, then $\mathcal{G}$ is connected.*

In addition, if $\delta(\mathbf{p}) > n_{max}$, then $\delta(\mathbf{p})$ is called reification index; if $\rho(\mathbf{p}) > n_{max}$, then $\rho(\mathbf{p})$ is called reification index.

Note that such a relation mapping is capable of joining more than one reification concepts to form a $\mathcal{NDL}$ relation.

```
**Input**: $\mathcal{SHOIN}$ DL knowledge base $\mathcal{K}$ with partially defined $\langle \nu, \delta, \rho \rangle$
**Output**: transformation parameter for $\mathcal{K}$
$W \leftarrow$ set of all property names in $K$
$X \leftarrow$ set of all property axioms in $K$
$Q_1, Q_2 \leftarrow \emptyset$ e  while  $W \neq \emptyset$ do
    if $Q_1 = Q_2 = \emptyset$ then
        $reverse = false$
        $p \leftarrow pop\_min(W)$
        if $\nu(p)$ is not defined  then
            $\nu(p) := p$ ; $\delta(p) := 1$ ; $\rho(p) := 2$

    if $Q_1 \neq \emptyset$ then
        $reverse = false$
        $p \leftarrow pop(Q_1)$
    else
        $reverse = true$
        $p \leftarrow pop(Q_2)$

    $A \leftarrow \{x \mid x \in X \wedge p$ occurs in $x\}$
    $X \leftarrow (X - A)$
    foreach $x \in A$ do
        $q \leftarrow$ atomic property different from $p$ occurring in $x$
        $\nu(q) := \nu(p)$
        if  ($p$ is accordant with $q$ in $x$) xor reverse then
            $Q_1 \leftarrow Q_1 \cup \{q\}$
            $\delta(q) := \rho(p)$ ; $\rho(q) := \delta(p)$
        else
            $Q_2 \leftarrow Q_2 \cup \{q\}$
            $\delta(q) := \delta(p)$ ; $\rho(q) := \rho(p)$

    end
    $W \leftarrow (W - \{p\})$
end
return  $\langle \nu, \delta, \rho \rangle$
```

Algorithm 1: Algorithm for defining well-formed transformation parameter

7 Comparison of reification algorithms

Great advantage of binary description logics such as $\mathcal{SHIQ}$ DL is reasoning support. Nowadays, the current of state-of-art of DL reasoners provide variety of reasoning services such as satisfiability of concepts or knowledge bases, subsumption hierarchies of concepts or roles, realization of individuals etc. Thanks to reification algorithms introduced in Chapter 5, the same set of services can be provided for $\mathcal{NDL}$. However, the reification algorithms generate a lot of very expressive axioms that might cause such services to be unusable in real applications. For further use of the algorithms, both theoretical and empirical analysis of reasoning over reified KBs need to be done. Asymptotic size and complexity of knowledge bases produced by the algorithms will be discussed in Section 7.1. In section Section 7.2, new framework for testing reasoner performance on real knowledge bases will be introduced. Data for testing will be gathered from existing owl ontologies that will be transformed via algorithms discussed in Chapter 6.

7.1 Theory-based comparison

Chapter 5 introduced 3 different reification algorithms. First algorithm (T-SHIQ) uses special DL concepts to represent $\mathcal{NDL}$ individuals, tuples and restrictions about them. Second algorithm (A-SHIQ) and third algorithm (A-SHIF) reifies $\mathcal{NDL}$ KB in more natural way because it uses DL individuals to represent $\mathcal{NDL}$ individuals and tuples.

Reification algorithms are described by two functions $\sigma(.)$ and $f(.)$ Function $\sigma(.)$ ensure correctness of the transformation, while function $f(.)$ adds additional axioms to the new knowledge base to ensure completeness of the transformation.

7.1.1 Proposed Metrics

Precise prediction of time that reasoner spends on evaluating some task is in most cases very hard or even impossible. Performance speed depends on implementation of the inference algorithms and various optimization techniques. Ontology cannot be split into simple elements which would be evaluated individually, because combinations of constructs is often subject of the optimizations. However, there are some assumptions that can be made. Performance slows down with growing number of axioms and its length. Also, some types of axioms are generally considered to cause performance loss more than the others.

In the theoretical comparison of this chapter, complexity of transformation will be evaluated in terms of axioms. Axioms of different kinds such as concept inclusion axioms, role inclusion axioms, concept or role assertions etc. will be distinguished. Also some parts of transformations generate variable length of axioms or variable number of axioms that depend on for example number of individuals of original knowledge base. Distinction between variable and constant dependency needs to be also considered in the final result.

7.1.2 Formal Notation

In following text, we use term complexity of axiom to refer to evaluation of type together with length of axiom. For each type of axiom there is a function that represent its complexity. Following different function exists:

- gci (for general concept inclusion axioms)
- sub_C (for primitive concept definitions / concept subsumption axioms)
- sub_R (for role inclusion axioms / role subsumption axioms)
- $assert_C$ (for concept assertion axioms)
- $assert_R$ (for role assertion axioms)
- $trans_R$ (for role transitivity axioms)

Each of the functions accepts one argument that is asymptotic length of axiom. Thus, for example expression $gci(1)$ describes complexity of general concept inclusion axiom of constant length and $assert_C(n)$ describes complexity of concept assertion axiom of length linearly dependent on argument n. Available variables that can be used for evaluation are:

- n_{max} - maximum arity of relations occurring in $\mathcal{NDL}$ KB
- $\#i$ - number of individuals occurring in $\mathcal{NDL}$ KB

- $\#t$ - number of tuples occurring in $\mathcal{NDL}$ KB
- $\#C_{atomic}$ - number of atomic concepts occurring in $\mathcal{NDL}$ KB
- $\#R_{atomic}$ - number of atomic relations occurring in $\mathcal{NDL}$ KB
- $\#C_{assertion}$ - number of concept assertions in $\mathcal{NDL}$ KB
- $\#R_{assertion}$ - number of relation assertions in $\mathcal{NDL}$ KB

Moreover, to denote complexity of axioms which are direct translation of schema $\mathcal{S}$ of some $\mathcal{NDL}$ KB, it is used notion $G(\mathcal{S})$.

Complexity evaluation based on above defined notation is too complicated to simply and intuitively differentiate transformations based on Calvanese and Horrocks. Therefore we define less precise metrics, the function $l(.)$. The function computes length of a concept as a number of all elementary concept and relation constructors from which the concept is built. Intuitively, $l(.)$ can be straightforwardly extended to relation, set of concepts, a set of relations, a set of axioms, or even whole knowledge base.

7.1.3 Complexity evaluation

To compare complexity of all approaches clearly, we have to evaluate complexity by as few different types of axioms as possible. For this reason some of the complex axiom are split into simplier ones, another ones are joined together to form one axiom of higher complexity. Following two rules are used to modify complexity of axioms in further text. The first, axiom $A \sqsubseteq B \sqcap C$ can be split into axioms $A \sqsubseteq B$ and $A \sqsubseteq C$. The second, axioms $A \sqsubseteq B$ and $C \sqsubseteq D$ can be joined together to form axiom $A \sqcup (C \sqcap \neg D) \sqsubseteq B \sqcap (\neg C \sqcup D)$. Note that the rules does not change intent of original axioms.

Direct translation of schema

Satisfiability-preserving function $\sigma(.)$ for terminological axioms of original knowledge base is same in each of the reification algorithms. In the transformation, every terminological axiom (i.e. concept or relation inclusion axiom) is translated to concept inclusion of the reified counterpart. Complexity of translated schema S for some $\mathcal{NDL}$ KB is $G(\mathcal{S})$.

Additional axioms for schema

Completeness of the schema is guaranteed in all of the approaches by the same axioms. Evaluation of their complexity is done as follows. Axiom

$\top \sqsubseteq \top_1 \sqcup ... \sqcup \top_{n_{max}}$ has complexity $sub_C(n_{max})$. Axiom $\top \sqsubseteq (\leq 1 f_1) \sqcap ... \sqcap (\leq 1 f_{n_{max}})$ is split into n_{max} axioms $\top \sqsubseteq (\leq 1 f_i)$ for $1 \leq i \leq n_{max}$, each of complexity $sub_C(1)$. On the other hand, general concept inclusion axioms $\forall f_i.\bot \sqsubseteq \forall f_{i+1}.\bot$ for $1 \leq i < n_{max}$ are joined together to form axiom $(\exists f_1.\top \sqcup \forall f_2.\bot) \sqcap ... \sqcap (\exists f_{n_{max}-1}.\top \sqcup \forall f_{n_{max}}.\bot) \sqsubseteq \top$. Length of the axiom depends linearly on n_{max} and hence the complexity is $gci(n_{max})$. Axioms $T_i \equiv \exists f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot$ for $2 \leq i \leq n_{max}$ are split into axioms $f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot \sqsubseteq T_i$ and $T_i \sqsubseteq f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot$, which is further divided into $i+1$ axioms $T_i \sqsubseteq \exists f_1.\top_1, ..., T_i \sqsubseteq \forall f_{i+1}.\bot$. Hence, the outcome complexity of the axioms is $(gci(n_{max}) + (\sum_{2 \leq i \leq n_{max}} (i + 1)) \times sub_C(1) = gci(n_{max}) + ({n_{max}}^2 - 3n_{max} + 2) \times sub_C(1)$. The outcome complexity of axioms $A \sqsubseteq \top_1$ and $A_{\mathbf{P}} \sqsubseteq \top_n$ for each atomic concept A and each atomic relation $\mathbf{P}$ is $\#C_{atomic} \times sub_C(1)$ and $\#R_{atomic} \times sub_C(1)$ respectively. Quick preview of above defined complexities is in Table 7.1. Finally, the outcome complexity of all additional axioms for schema is $2 \times gci(n_{max}) + sub_C(n_{max}) + (\#C_{atomic} + \#R_{atomic} + {n_{max}}^2 - 2n_{max} + 2) \times sub_C(1)$.

$f(\mathcal{S})$	Complexity
$\top \sqsubseteq \top_1 \sqcup ... \sqcup \top_{n_{max}}$	$sub_C(n_{max})$
$\top \sqsubseteq (\leq 1 f_1) \sqcap ... \sqcap (\leq 1 f_{n_{max}})$	$n_{max} \times sub_C(1)$
$\forall f_i.\bot \sqsubseteq \forall f_{i+1}.\bot$	$gci(n_{max})$
$T_i \equiv \exists f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot$	$({n_{max}}^2 - 3n_{max} + 2) \times sub_C(1) +$ $+ gci(n_{max})$
$A \sqsubseteq \top_1$	$\#C_{atomic} \times sub_C(1)$
$A_{\mathbf{P}} \sqsubseteq \top_n$	$\#R_{atomic} \times sub_C(1)$

Table 7.1: Quick preview of complexity of additional axioms for schema

Direct translation of *ABox*

Direct translation of $\mathcal{NDL}$ concept assertions and relation assertions is the same for A-SHIQ and A-SHIF. Length of axioms $A_w : \sigma(C)$ is dependent on $|\sigma(C)|$, which is directly proportional to $|C|$. Hence, the outcome complexity of axioms is $\sum_{w:C \in \mathcal{A}} assert_C(|C|)$. Similarly, axioms $t_{\vec{w}} : \sigma(R)$ have outcome complexity $\sum_{\vec{w}:\mathbf{R} \in \mathcal{A}} assert_C(|\mathbf{R}|)$. Axioms $\langle t_{\vec{w}}, w_i \rangle : f_i$ generate at most n_{max} axioms for each tuple occurring in $\mathcal{NDL}$ KB, thus their complexity is at most $n_{max} \times \#t \times assert_R(1)$.

For the T-SHIQ approach, the complexity of axioms $A_w \sqsubseteq \sigma(C)$ and $T_{\vec{w}} \sqsubseteq \sigma(\mathbf{R})$ is $\sum_{w:C \in \mathcal{A}} sub_C(|C|)$ and $\sum_{\vec{w}:\mathbf{R} \in \mathcal{A}} sub_C(|\mathbf{R}|)$ respectively. Axioms $T_{\vec{w}} \equiv \exists f_1.A_{w_1} \sqcap ... \sqcap \exists f_n.A_{w_n} \sqcap \forall f_{n+1}.\bot$ are split into general concept inclusion axioms and primitive concept definition axioms of constant length

The outcome complexity is $\#t \times gci(n_{max}) + n_{max} \times \#t \times sub_C(1)$. The outcome complexity of axioms $A_{w_i} \sqsubseteq \exists f_i^-.T_{\vec{w}} \sqcup \leq 1 f_i^-.T_{\vec{w}}$ is $n_{max} \times \#t \times sub_C(1)$. Quick preview of above defined complexities is in Table 7.2. Finally, the outcome complexity of direct translation of *ABox* for both A-SHIQ and A-SHIF algorithms is $\sum_{w:C\in\mathcal{A}} assert_C(|C|) + \sum_{\vec{w}:\mathbf{R}\in\mathcal{A}} assert_C(|\mathbf{R}|) + n_{max} \times \#t \times assert_R(1)$ and for T-SHIQ it is $\sum_{w:C\in\mathcal{A}} sub_C(|C|) + \sum_{\vec{w}:\mathbf{R}\in\mathcal{A}} sub_C(|\mathbf{R}|) + \#t \times (gci(n_{max}) + n_{max} \times sub_C(1))$.

	$\sigma(\mathcal{A})$	Complexity
A-SHIF/Q	$A_w : \sigma(C)$	$\sum_{w:C\in\mathcal{A}} assert_C(\lvert C\rvert)$
	$t_{\vec{w}} : \sigma(R)$	$\sum_{\vec{w}:\mathbf{R}\in\mathcal{A}} assert_C(\lvert\mathbf{R}\rvert)$
	$\langle t_{\vec{w}}, w_i\rangle : f_i$	$n_{max} \times \#t \times assert_R(1)$
T-SHIQ	$A_w \sqsubseteq \sigma(C)$	$\sum_{w:C\in\mathcal{A}} assert_C(\lvert C\rvert)$
	$T_{\vec{w}} \sqsubseteq \sigma(\mathbf{R})$	$\sum_{\vec{w}:\mathbf{R}\in\mathcal{A}} assert_C(\lvert\mathbf{R}\rvert)$
	$T_{\vec{w}} \equiv \exists f_1.A_{w_1} \sqcap ... \sqcap \exists f_n.A_{w_n} \sqcap \forall f_{n+1}.\bot$	$\#t \times gci(n_{max}) +$ $+ n_{max} \times \#t \times sub_C(1)$
	$A_{w_i} \sqsubseteq \exists f_i^-.T_{\vec{w}} \sqcup \leq 1 f_i^-.T_{\vec{w}}$	$n_{max} \times \#t \times sub_C(1)$

Table 7.2: Quick preview of complexity of direct translation of *ABox*

Additional axioms for *ABox*

Additional axioms for *ABox* in A-SHIQ approach consist of two different types of axioms. The first, axioms of type $w : Q$ with outcome complexity $\#i \times assert_C(1)$. The second, axioms of type $w_1 :\leq f_1^-(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n})$ have outcome complexity $\#t \times assert_C(n_{max})$.

In the A-SHIF approach complexity of $w : Q$ is computed same. Complexity of $g_{\vec{w}} \sqsubseteq f_1$ and $\top \sqsubseteq \leq 1 g_{\vec{w}}^-$ is $\#t \times sub_R(1)$ and $\#t \times sub_C(1)$ respectively. Finally, $\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap f_n.Q_{w_n} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1}$ generates complexity $\#t \times gci(n_{max})$.

In the T-SHIQ approach axiom $s_{root} : \exists create.A_{w_1} \sqcap ... \sqcap \exists create.A_{w_m}$ is split into $\#i + \#t$ concept assertions, each of complexity $assert_C(1)$. Each of the axioms $f_i \sqsubseteq U$, $f_i^- \sqsubseteq U$, $create \sqsubseteq U$ $create^- \sqsubseteq U$ has complexity $sub_R(1)$ and $trans(U)$ has complexity $trans_R(1)$. Finally, axioms of type $(A \sqcap C) \sqsubseteq \forall U.(\neg A \sqcup C)$ for $A \in \mathcal{A}_{\mathcal{S}'}, C \in CL'(\mathcal{S}')$ are dependent on size of $\mathcal{A}_{\mathcal{S}'}$ and size of extended Fischer-Ladner closure of partially reified schema $\mathcal{S}'$, thus $CL'(\mathcal{S}')$. The size of $\mathcal{A}_{\mathcal{S}'}$ is equal to number of representative concepts therefore it is $\#i + \#t$. Outcome complexity of the axioms is $(\#i + \#t) \times \mathcal{O}(n_{max} \times l(\mathcal{K})) \times gci(n_{max} \times l(\mathcal{K}))$. Detailed explanation of this complexity is explained further in text.

Quick preview of above defined complexities is in Table 7.3.

	$f(\mathcal{A})$	Complexity
A-SHIQ	$w : Q$	$\#i \times assert_C(1)$
	$w_1 :\leq f_1^-(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n})$	$\#t \times assert_C(n_{max})$
A-SHIF	$w : Q$	$\#i \times assert_C(1)$
	$g_{\vec{w}} \sqsubseteq f_1$	$\#t \times sub_R(1)$
	$\top \sqsubseteq \leq 1 g_{\vec{w}}^-$	$\#t \times sub_C(1)$
	$\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap f_n.Q_{w_n} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1}$	$\#t \times gci(n_{max})$
T-SHIQ	$s_{root} : \exists create.A_{w_1} \sqcap ... \sqcap \exists create.A_{w_m}$	$\#i \times assert_C(1)$
	$f_i \sqsubseteq U, f_i^- \sqsubseteq U$	$2 \times n_{max} \times sub_R(1)$
	$create \sqsubseteq U, create^- \sqsubseteq U$	$2 \times sub_R(1)$
	$trans(U)$	$trans_R(1)$
	$(A \sqcap C) \sqsubseteq \forall U.(\neg A \sqcup C)$	$(\#i + \#t) \times \mathcal{O}(n_{max} \times l(\mathcal{K}))$ $\times gci(n_{max} \times l(\mathcal{K}))$

Table 7.3: Quick preview of complexity of additional axioms for *ABox*

Extended Fischer-Ladner closure

In the following text, the complexity of extended Fischer-Ladner closure of T-SHIQ reified counterpart based on complexity of original $\mathcal{NDL}$ KB is evaluated.

Let $\mathcal{K} = \langle \mathcal{S}, \mathcal{A} \rangle$ be a $\mathcal{NDL}$ knowledge base. From $\mathcal{K}$, the complexity of extended Fischer-Ladner closure $CL'(S')$ as defined in Construction 5.2.1 can be evaluated as follows.

- $l(\mathcal{K}) = l(\mathcal{S}) + l(\mathcal{A})$
- $l(\sigma(\mathcal{S})) = \mathcal{O}(l(\mathcal{S}))$; In $f(\mathcal{S})$ axioms $T_i \equiv \exists f_1.\top_1 \sqcap ... \sqcap \exists f_i.\top_1 \sqcap \forall f_{i+1}.\bot$ generate length $n_{max}{}^2$, however, if there are no relations of some arity j, axiom for T_j does not have to be included in reified counterpart. Thus, $l(f(\mathcal{S})) = \mathcal{O}(l(\mathcal{S}) + n_{max} \times l(\mathcal{S}))$; In $\sigma'(\mathcal{A})$ each of the tuple assertions is reified to at most n_{max} terminological axioms, thus $l(\sigma'(\mathcal{A})) = \mathcal{O}(n_{max} \times l(\mathcal{A}))$
- $\mathcal{S}'$ is defined as $\sigma(\mathcal{S}) \cup f(\mathcal{S}) \cup \sigma'(\mathcal{A})$, thus $l(\mathcal{S}') = l(\sigma(\mathcal{S})) + l(f(\mathcal{S})) + l(\sigma'(\mathcal{A})) = \mathcal{O}(2 \times l(S) + n_{max} \times l(\mathcal{S}) + n_{max} \times l(\mathcal{A}))$. Hence $l(\mathcal{S}') = \mathcal{O}(n_{max} \times l(\mathcal{K}))$.
- In the definition of extended Fischer-Ladner closure (Definition 5.2.2): first, $\mathcal{S}'$ is translated to $C_{\mathcal{S}'} = \bigsqcap_{A \sqsubseteq B \in \mathcal{S}'} (\neg A \sqcup B)$.Hence, $l(C_{\mathcal{S}'}) = l(\mathcal{S}') = \mathcal{O}(n_{max} \times l(\mathcal{K}))$; second, the $CL(C_{\mathcal{S}'})$ is computed. The size of set $CL(C_{\mathcal{S}'})$ is linearly bounded by size of $C_{\mathcal{S}'}$ [49]. Note that this is only upper bound of complexity, however, it is easy to show that lower

bound is the same. Thus $CL(C_{\mathcal{S}'})$ contains $\mathcal{O}(n_{max} \times l(\mathcal{K}))$ concepts, each of the length $\mathcal{O}(n_{max} \times l(\mathcal{K}))$; third, $CL'(\mathcal{S}')$ increases size of the $CL(C_{\mathcal{S}'})$ and length of each concept from the $CL(C_{\mathcal{S}'})$ only linearly.

Hence, $CL'(\mathcal{S}')$ produces at most $\mathcal{O}(n_{max} \times l(\mathcal{K}))$ concepts, each of the length at most $\mathcal{O}(n_{max} \times l(\mathcal{K}))$.

Axioms of type $(A \sqcap C) \sqsubseteq \forall U.(\neg A \sqcup C)$ are in T-SHIQ algorithm generated for each representative concept A and each $C \in CL'(\mathcal{S}')$. The number of representative concepts is #i+#t. Each of the axioms is general concept inclusion axiom of length directly proportional to $|C|$. Hence, the complexity of axioms used for singling out representative concept is :

$$\mathcal{O}(n_{max} \times l(\mathcal{K})) \times (\#i + \#t)) \times gci(n_{max} \times l(\mathcal{K})).$$

Comparison of A-SHIQ and A-SHIF

Reification algorithms A-SHIQ and A-SHIF differ only in definition of additional axioms that axiomatize tuple-admissibility of *ABox*. According to A-SHIQ reification algorithm, additional axioms for *ABox* have outcome complexity $\#i \times assert_C(1) + \#t \times assert_C(n_{max})$. In the A-SHIF reification, the outcome complexity of *ABox* additional axioms is $\#i \times assert_C(1) + \#t \times (gci(n_{max}) + sub_C(1) + sub_R(1))$.

To simplify the comparison of complexity we will modify both transformations for the last time. These modifications add some complexity to both of the algorithms. However, the increment of the complexity is the same for both algorithms and therefore it will not affect the final comparison of A-SHIQ and A-SHIF.

The modifications are done as follows. We introduce new concepts $X_{\vec{w}}$ for each tuple $\vec{w}$ occurring in original $\mathcal{NDL}$ KB and add to both transformations axioms $(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n}) \sqsubseteq X_{\vec{w}}$. Then, in A-SHIQ transformation, axioms of type $w_1 :\leq f_1^-(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n})$ are substituted by axioms $w_1 : X_{\vec{w}}$ and in A-SHIF transformation, axioms of type $\top_n \sqcap \exists f_2.Q_{w_2} \sqcap ... \sqcap f_n.Q_{w_n} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1}$ are substituted by axioms $X_{\vec{w}} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1}$. Note that these modification of transformations did not change satisfiability of KBs.

In the modified transformations, axioms $(\top_n \sqcap \exists f_2.Q_{w_1} \sqcap ... \sqcap \exists f_n.Q_{w_n}) \sqsubseteq X_{\vec{w}}$ have outcome complexity $\#t \times gci(n_{max})$. Axioms $w_1 : X_{\vec{w}}$ and $X_{\vec{w}} \sqsubseteq \exists g_{\vec{w}}.Q_{w_1}$ have outcome complexity $\#t \times assert_C(1)$ and $\#t \times sub_C(1)$. Hence, the outcome complexity of new A-SHIQ algorithm is $\#i \times assert_C(1) + \#t \times (gci(n_{max}) + assert_C(1))$, while in new A-SHIF algorithm it is $\#i \times assert_C(1) + \#t \times (gci(n_{max}) + sub_C(1) + sub_R(1))$.

Thus, in addition to A-SHIF, A-SHIQ contains $\#t \times assert_C(1)$ axioms but lacks $\#t \times (sub_C(1) + sub_R(1))$ axioms.

If we omit types of axioms length of reified counterpart is $\mathcal{O}(l(\mathcal{K})+n_{max}\times \#t)$, thus at most $\mathcal{O}(n_{max}\times l(\mathcal{K}))$.

7.1.4 Summary

In theoretical comparison of complexity we have substituted some axioms for their equivalents in order to get as simple comparison results as possible.

In addition to A-SHIF, A-SHIQ generates one concept assertion of constant length for each tuple, but lucks one primitive concept definition of constant length and role subsumption axiom for each tuple.

If $\mathcal{K}$ is an $\mathcal{NDL}$ KB then length of reified counterpart of $\mathcal{K}$ grows at least with square of $l(\mathcal{K})$. Moreover we showed that it is at most $\mathcal{O}((\#i \times \#t) \times (n_{max} \times l(\mathcal{K}))^2)$, thus $\mathcal{O}({n_{max}}^2 \times l(\mathcal{K})^3)$.

Length of reified counterpart of A-SHIQ and A-SHIF is at most $\mathcal{O}(l(\mathcal{K})+n_{max} \times \#t)$, thus $\mathcal{O}(n_{max} \times l(\mathcal{K}))$.

7.2 Empirical comparison

7.2.1 Tested data

Testing reasoner performance on reified knowledge bases needs real $\mathcal{NDL}$ KBs to be created. We used real, well known OWL ontologies and transformed them into $\mathcal{NDL}$ KBs by algorihms discussed in Chapter 6. For some of the ontologies two or more object properties could be joined together to form $\mathcal{NDL}$ relation of higher arity than 2. For the others only binary transformation was semantically suitable. Binary transformations were done by automatic algorithm from Chapter 6. For transformation with relations of higher arity, mappings of new relations had to be created to supply semiautomatic algorithm.

Reification algorithms discussed in Chapter 5 differs only in transformation of *ABox* axioms and therefore for comparison, ontologies with asserted facts are needed. However, most of the ontologies consist of terminological axioms only. To overcome this problem, KAON OWL tools were used to randomly populate some of the ontologies without individuals. For the comparison we have used the following knowledge bases:

- **Deidre** is example knowledge base about Deidre Capron from Chapter 4. It contains two ternary relations **come** and **work**.
- **Pizza** knowledge base was created from an example ontology that contains all constructs required for the Pizza Tutorial run by Manch-

ester University [56]. The original ontology contains object properties **hasBase** and **hasTopping** that relate individuals of class *Pizza* to classes *PizzaBase* and *PizzaTopping* respectively. We have joined these object properties to form one ternary relation **pizzaProperties** and added some tuples of this relation to the KB.

- **Generations** knowledge base was based on ontology "Generations" [57] that was designed to help describe family relationships with use of reasoner. The knowledge base already contained significant amount of individuals and it is direct product of automatic transformation discussed in Chapter 6.

7.2.2 Target systems

We have chosen three of the current of the state-of-art of reasoners. In the following text details about the reasoners are explained.

Racer

RACER stands for Renamed *ABox* and Concept Expression Reasoner. RacerPro is its commercial name. Beta version of RacerPro 1.9.2, that was used in this comparison, was released on 24/10/2007. It implements highly optimized tableau calculus for description logics $\mathcal{ALCQHI}_{\mathcal{R}+}$ also known as $\mathcal{SHIQ}$as well as following optimization techniques: dependency-directed backtracking and DPLL-style semantic branching, transformation of axioms (GCIs), model caching and model merging. Except for two limitations, RacerPro supports OWL-DL completely. First, individuals in class expressions (i.e. nominals) are only approximated. Second, although all required datatypes of OWL-DL are supported, RacerPro cannot process user-defined datatype types given as external XML Schema specification. Since in the reification algorithms there are neither nominals nor datatypes, RacerPro is capable of reasoning over reified $\mathcal{NDL}$ knowledge bases.

Basic interface for controlling the reasoner is so-called KRSS-based interface. It is based on lisp-like language that they call Mini-Lisp. Main purpose of this interface was to directly present the declaration and results of queries in a brief and human-readable form.

Java programmers can benefit from JRacer, which is the client library to access the services of a RacerPro server through Java API. However, the API is very simple and clumsy. Construction of the reasoner request has to be done without API as it is a string in Mini-Lisp format.

Second, RacerPro offers to communicate with the reasoner server by the HTTP-based DIG protocol. The DIG protocol is less expressive than the native one. Currently, RacerPro supports DIG 1.1 completely and DIG 2.0 partially [55].

Pellet

Pellet is an open source, OWL-DL reasoner in Java that is developed, and commercially supported, by Clark&Parsia, LLC. For the comparison, Pellet 1.5.1 released on 26/10/2007 was used. It is based on tableaux algorithms for expressive DLs and supports full expressivity of OWL-DL as well as all the features proposed in OWL 1.1, with exception of n-ary datatypes. It implements TBox partitioning, absorption and lazy unfolding plus dependency directed backjumping, semantic branching and early blocking strategies.

Pellet provide many interfaces to the reasoner. For Java programmers, there is direct in-memory implementation of the reasoner interfaces Jena and Manchester OWL-API library. Also Pellet can act as a server with the support of DIG 1.1 protocol.

FaCT++

FaCT++ is a new generation of the well-known FaCT OWL-DL reasoner. It implements a new tableaux decision procedure for $\mathcal{SHOIQ}$ [29]. For the comparison, FaCT 1.1.10 released on 30/10/2007 was used.

FaCT++ provide support for direct in-memory implementation for Manchester OWL-API. It can also be started as DL reasoner with DIG 1.1 protocol support.

7.2.3 Target reasoning queries

The reasoners in the comparison were queried for three different tasks :

- *Consistency checking* finds out if an ontology does not contain any contradictory axioms, i.e. if the ontology is satisfiable.
- *Classification* computes the subclass relations between every named (atomic) classes and create the complete class hierarchy. The class hierarchy allows to answer queries such as getting all or only the direct subclasses of a class.
- *Realization* finds out the most specific classes (i.e. direct types) for each individual from ontology. Realization triggers classification since direct

types are defined with respect to a class hierarchy. With classification hierarchy, it is possible to get all types for each individual from the ontology.

7.2.4 Testing configuration

Benchmark was done on a Linux box featuring an Intel® Celeron® CPU at 3,2 GHz and 2 GB main memory. For saving and loading reified counterparts we have chosen OWL1.1 ontologies that were parsed by OWL1.1 API [54]. Note that OWL-DL ontologies could not be used for this purpose because OWL-DL does not support qualified number restrictions.

To achieve fair conditions for all reasoners we had to choose same interface for querying them. For this purpose, the DIG interface [53] was chosen. The interface is intended to provide uniform access to description logics reasoners. It defines a simple protocol (based on HTTP PUT/GET) along with an XML Schema that describes a concept language and accompanying operations.

Each of the reasoners that were used in benchmark support at least DIG 1.1 protocol. DIG 1.1 is capable to expressing queries in $\mathcal{SHOIQ}$ DL with some additional support for datatypes [50].

The benchmark is iteration of following steps:

1. Start one of the reasoners as a DIG 1.1 server.
2. Load next ontology via OWL1.1 API to the memory.
3. Send the ontology via http protocol to the server.
4. Send one of the possible queries (consistency check, classification, realization) to the server and receive the answer. This time is measured.
5. Cancel running of the reasoner server.

Note that at the beginning of each iteration new reasoner server was started (*1.*) and at the end, it was canceled (*5.*). It was done to prevent reasoners from caching knowledge bases. Also note that in one iteration we answered to only one of three possible simple queries.

A time limit of one hour (3600 seconds) was set for one query, meaning that any process was aborted if its computation time exceeds 3600 seconds. Each query of one ontology was executed 10 times and the average of the values was stated to be final result.

7.2.5 Results

Results of the benchmark are summarized in following tables. Each of the tables describes one query for each reified counterpart of all ontologies and all reasoners. In addition to consistency check, classification and realization, the loading times were also evaluated. If reasoner did not finish some of the tasks in one hour or run out of memory, it is displayed by ∞ in the tables.

The Table 7.4 shows that ontologies of T-SHIQ approach were loaded from 30 to 100 times slower than the others. FaCT++ was fastest reasoner for loading most of the reified ontologies. Pellet finished in all of the cases as the last.

	ontology\reasoner	Racer	Pellet	FaCT++
T-SHIQ	Deidre	3409.51	4712.12	1903.12
	Pizza	94078.21	110340.56	69931.10
	Generations	23072.83	36053.11	17051.21
A-SHIQ	Deidre	85.92	135.06	33.45
	Pizza	1011.72	3957.71	566.12
	Generations	141.45	255.41	64.5
A-SHIF	Deidre	28.28	132.84	33.89
	Pizza	1021.54	4025.55	625.78
	Generations	158.46	303.93	66.79

Table 7.4: Loading times of reified counterparts in miliseconds

	ontology\reasoner	Racer	Pellet	FaCT++
T-SHIQ	Deidre	991.87	∞	∞
	Pizza	∞	∞	∞
	Generations	∞	∞	∞
A-SHIQ	Deidre	13.65	117.35	46.86
	Pizza	1002.32	8130.96	1539.46
	Generations	108.28	339.76	164.04
A-SHIF	Deidre	14.24	76.98	47.41
	Pizza	997.63	130103.37	1420.31
	Generations	20.06	465.46	83.79

Table 7.5: Consistency check times of reified counterparts in miliseconds

	ontology\reasoner	Racer	Pellet	FaCT++
T-SHIQ	Deidre	∞	∞	∞
	Pizza	∞	∞	∞
	Generations	∞	∞	∞
A-SHIQ	Deidre	81.98	379.65	70.73
	Pizza	1819.23	19786.53	2085.57
	Generations	167.73	613.68	176.56
A-SHIF	Deidre	79.09	345.52	72.62
	Pizza	1888.21	137490.1	1920.31
	Generations	109.96	808.82	102.8

Table 7.6: Classification times of reified counterparts in miliseconds

	ontology\reasoner	Racer	Pellet	FaCT++
T-SHIQ	Deidre	∞	∞	∞
	Pizza	∞	∞	∞
	Generations	∞	∞	∞
A-SHIQ	Deidre	59.6	357.89	50.03
	Pizza	1692.15	29843.98	1499.12
	Generations	604.9	5258.39	209.63
A-SHIF	Deidre	58.79	285.49	57.3
	Pizza	1501.2	131500.96	1470.33
	Generations	445.94	868.5	91.33

Table 7.7: Consistency check times of reified counterparts in miliseconds

From the other tables it is obvious, that the fastest reasoner for the reified ontologies is Racer, next FaCT++ and the last is Pellet. Approach T-SHIQ is unusable for reasoning, since on even small ontology such as Deidre reasoning task did not finish in all of the reasoners. From approaches A-SHIF and A-SHIQ, reasoning tasks in A-SHIF were computed faster in most of the cases.

8 Summary and discussion

What are pros and cons of state-of-art Knowledge Representation Systems with support for n-ary relations ? What are drawbacks of representing n-ary relations by binary ones ? How can be reasoning support added to language with n-ary relations without having to reimplement reasoning algorithms ? These are the basic questions that have been addressed in this thesis.

In first chapter the importance together with general problems of n-ary relations were outlined. Theoretical background of this thesis is described in chapter two.

In chapter four, based on existing language $\mathcal{DLR}$, new language $\mathcal{NDL}$ with support for n-ary relations is designed. Semantics and abstract syntax of the language is very similar to $\mathcal{DLR}$ and therefore only differences are explained. New graphical notation is also provided.

Chapter five describes three different algorithms that can be used for $\mathcal{NDL}$ to transform into description logics within expressivity $\mathcal{SHIQ}$. These transformations (also called reification algorithms) preserve satisfiability and hence satisfiability check of $\mathcal{NDL}$ knowledge base can be provided by its transformed counterpart. First of the algorithms is based on Calvanese work in which $\mathcal{DLR}$ was transformed into logics $\mathcal{CIQ}$ DL. Due to different expressivity of the languages $\mathcal{CIQ}$ DL and $\mathcal{SHIQ}$ DL, some constructs needed to be translated to new ones. The chapter also contains proof that new constructs are correct. Second and third of the algorithms is based on Horrocks work in which $\mathcal{DLR}$ was transformed into $\mathcal{SHIQ}$ DL.

Transformations of binary description logics into $\mathcal{NDL}$ is discussed in chapter six. It contains automatic algorithm for translating $\mathcal{SHOIQ}$ DL knowledge base into binary $\mathcal{NDL}$ knowledge base. For translating $\mathcal{SHOIQ}$ DL KB into $\mathcal{NDL}$ KB with relations of higher arity, semi-automatic algorithm is described.

Thanks to algorithms presented in chapters five and six we have framework for translating $\mathcal{NDL}$ from and to $\mathcal{SHIQ}$ DL. With little work using this framework $\mathcal{NDL}$ can provide same support for reasoning as it is available for $\mathcal{SHIQ}$ DL. However, also complexity of the algorithms is an issue. There would be no use of reasoning services that are too complex. For this reason, chapter seven is dedicated to evaluate complexity of all three reification transformations. At first complexity comparison of reified knowledge bases is evaluated by theoretical analysis of axioms that reification algorithms generate. Then comparison is done on real $\mathcal{NDL}$ knowledge bases with use of the current state-of-art of reasoners for binary description logics.

Bibliography

[1] F. Manola and E. Miller: RDF Primer, W3C Recommendation, February 2004,
http://www.w3.org/TR/rdf-primer/.

[2] D. Brickley and R. Guha: RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, February 2004, http://www.w3.org/tr/rdf-schema/.

[3] D. Beckett and B. McBride: RDF/XML syntax specification, W3C Recommendation, February 2004,
http://www.w3.org/TR/rdf-syntax-grammar/.

[4] M. Dean and G. Schreiber: OWL Web Ontology Language Reference, W3C Recommendation, February 2004,
http://www.w3.org/tr/owl-ref/.

[5] T. Bray, J. Paoli, and C. M. Sperberg-McQueen: Extensible markup language (xml) volume 2, pp. 27-66, 1997.

[6] N. Noy, A. Rector: Defining N-ary Relations on the Semantic Web, W3C Working Group Note 12, April 2006,
http://www.w3.org/TR/swbp-n-aryRelations/.

[7] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks: OWL web ontology language semantics and abstract syntax, W3C Recommendation, 2004,
http://www.w3.org/TR/owl-semantics/.

[8] Project CIPHER (Communities of Interest Promoting Heritage of European Regions):
http://cipherweb.open.ac.uk/.

[9] N. Guarino, P. Giaretta: Ontologies and knowledge bases (Towards a Terminological Clarification), Large knowledge bases: Knowledge Building and Knowledge Sharing 1995. IOS Press, Amsterdam: 25-32.

[10] M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf: Terminological systemsrevisited: Terminology = schema + views, In Proc.of the 12th Nat. Conf. on Articial Intelligence (AAAI-94), pp. 199-204, Seattle, USA, 1994.

[11] A. Y. Levy and M. Rousset: CARIN: A representation language combining Horn rules and description logics, In Proc. of the 12th European Conf. on Articial Intelligence (ECAI-96), pp. 323-327, 1996.

[12] J. F. Sowa: Conceptual Structures: Information Processing in Mind and Machine, Addison Wesley, 1984,

[13] M. Niinimäki: Understanding the Semantics of Conceptual Graphs, 1999.

[14] J.-F. Baget and M.-L. Mugnier: The complexity of rules and constraints, JAIR 16 (2002), pp. 425-465.

[15] E. Kilfeather: Enabling Communities of Interest to Promote Heritage of European Regions, Dublin Institute of Technology, September 2004.

[16] J. F. Sowa: Conceptual Graphs Summary, in: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.), Conceptual Structures: current research and practice, Ellis Horwood, 1992, 3-51.

[17] C. S. Pierce, J. F. Sowa: Existential Graphs, MS 514 by Charles Sanders Pierce with commentary by John F. Sowa.

[18] G. Kerdiles and E. Salvat: A sound and complete proof procedure based on tableaux and projection, In Proc. of ICCS 97, LNAI 1257, pp. 371-385, Springer Verlag, 1997.

[19] M.-L. Mugnier, M. LeclÃÍre: On querying simple conceptual graphs with negation, 2007.

[20] J. F. Sowa: Conceptual Graphs: Draft Proposed American National Standard, ICCS'99 Proc., LNAI 1640, pp. 1-65, Springer Verlag, 1999.

[21] J. Esch, R. Levinson: An Implementation Model for Contexts and Negation in Conceptual Graphs, International Conference on Conceptual Structures, 1995.

[22] K. Matoušek, L. Král, and M. Falc: Apollo CH Manual, August 2004, http://krizik.felk.cvut.cz/cipher/manuals/Apollo_manual.PDF.

[23] F. Baader and W. Nutt: Basic description logics, In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, The Description Logic Handbook: Theory, Implementation, and Applications, pp. 43-95. Cambridge University Press, 2003.

[24] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[25] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi: Reasoning in expressive description logics, In Alan Robinson and Andrei Voronkov, editors, Handbook of Automated Reasoning, pp. 1581-1634, Elsevier Science Publishers, 2001.

[26] I. Horrocks: Using an expressive description logic: Fact or fiction?, In Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98), pp. 636-647, 1998.

[27] S. Tobies: Complexity Results and Practical Algorithms for Logics in Knowledge Representation, Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[28] I. Horrocks and P. Patel-Schneider: Reducing OWL entailment to description logic satisfiability, J. of Web Semantics, 1(4):345-357, 2004.

[29] Horrocks, I., Sattler, U.: A tableaux decision procedure for SHOIQ, In Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), 2005, To appear.

[30] Baader, F., Hanschke, P.: A schema for integrating concrete domains into concept languages, In Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI.91), pp. 452-457, 1991.

[31] De Giacomo, G.: Decidability of Class-Based Knowledge Representation Formalisms, Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma, 1995.

[32] Horrocks, I. and U. Sattler: Ontology Reasoning in the SHOQ(D) Description Logic, In Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), pp. 199-204, 2001.

[33] Hladik, J. and J. Model: Tableau Systems for SHIO and SHIQ, In Proc. of the 2004 Description Logic Workshop (DL 2004), CEUR, http://ceur-ws.org, 2004

[34] T. Catarci and M. Lenzerini: Representing and using interschema knowledge in cooperative information systems, Journal of Intelligent and Cooperative Information Systems, pp. 375-398, 1993.

[35] G. De Giacomo and M. Lenzerini: Description logics with inverse roles, functional restrictions, and n-ary relations, In Proc. of the 4th European Workshop on Logics in Articial Intelligence (JELIA-94), LNCS vol. 838, pp. 332-346, Springer-Verlag, 1994.

[36] G. De Giacomo and M. Lenzerini: What's in an aggregate: Foundations for description logics with tuples and sets, In Proc. of the 14th Int. Joint Conf. on Articial Intelligence (IJCAI-95), pp. 801-807, 1995.

[37] D. Calvanese, G. De Giacomo, and M. Lenzerini: Structured objects: Modeling and reasoning, In Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD-95), LNCS vol. 1013, pp. 229-246. Springer-Verlag, 1995.

[38] D. Calvanese, G. De Giacomo, M. Lenzerini: Conjunctive Query Containment in Description Logics with n-ary Relations, 1997.

[39] D. Calvanese, G. De Giacomo, and M. Lenzerini: On the decidability of query containment under constraints, In Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS98), pp. 149-158, 1998.

[40] I. Horrocks, U. Sattler, and S. Tobies: Practical reasoning for expressive description logics, In H. Ganzinger, D. McAllester, and A. Voronkov, editors, Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR99), pp. 161-180, 1999.

[41] I. Horrocks: FaCT and iFaCT, In Proceedings of the International Workshop on Description Logics (DL99), pp. 133-135, 1999.

[42] M. Jarke, C. Quix, D. Calvanese, M. Lenzerini, E. Franconi, S. Ligoudistiano, P. Vassiliadis, and Y. Vassiliou: Concept based design of data warehouses: The DWQ demonstrators, In 2000 ACM SIGMOD Intl. Conference on Management of Data, 2000.

[43] E. Franconi and G. Ng: The ICOM tool for intelligent conceptual modelling, In Proc. of the 7th International Workshop on Knowledge Representation meets Databases (KRDB2000), 2000.

[44] I. Horrocks, U. Sattler, S. Tessaris, S. Tobies: How to decide Query Containment under Constraints using a Description Logic(1999), Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'2000), 2000.

[45] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati: Use of the data reconciliation tool at telecom italia, DWQ deliverable D.4.3, Foundations of Data Warehouse Quality (DWQ), 1999.

[46] D. Calvanese and M. Lenzerini: TBox and ABox Reasoning in Expressive Description Logics, 1996.

[47] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies: Query containment using a DLR ABox, LTCS-Report 99-15, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1999.

[48] D. Kozen and J. Tiuryn: Logics of programs, In Handbook of Theoretical Computer Science Formal Models and Semantics, J. van Leeuwen, Ed. Elsevier Science Publishers, pp. 789-840, 1990.

[49] M. J. Fischer and R. E. Ladner: Propositional Dynamic Logic of Regular Programs, JCSS, 18, 194-211, 1979.

[50] S. Bechhofer: The DIG Description Logic Interface: DIG/1.1, Proceedings of DL2003 Workshop, Rome, June 2003.

[51] A. Kalyanpur: Debugging and Repair of OWL Ontologies, Ph.D. Dissertation, University of Maryland College Park, 2006.

[52] A Hypermedia-based Featherweight OWL Ontology Editor,
http://www.mindswap.org/2004/SWOOP/.

[53] The DIG interface standard,
http://www.dl.kr.org/dig/.

[54] The OWL 1.1 API,
http://owlapi.sourceforge.net/.

[55] RacerPro Users Guide Version 1.9.2, Racer Systems GmbH & Co. KG, October 18, 2007,
http://www.racer-systems.com.

[56] Pizza ontology:
http://www.co-ode.org/resources/tutorials/.

[57] Generations ontology:
http://protege.cim3.net/file/pub/ontologies/generations/generations.owl