2012

Enhancing Description Logics for Rules Coverage

David Carral Martinez
Wright State University

Enhancing Description Logics For Rules Coverage

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science

By

DAVID CARRAL MARTÍNEZ
B.S., Universidad Pontificia de Salamanca, 2011

2012
Wright State University
Ohio Center of Excellence in Knowledge-enabled Computing

WRIGHT STATE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

June 30, 2010

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY `David Carral Martínez` ENTITLED `Enhancing Description Logic for Rules Coverage` BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF `Master of Science`.

Pascal Hitzler, Ph. D.
Thesis Director

Mateen Rizki, Ph.D.
Department Chair

Committee on
Final Examination

Pascal Hitzler, Ph. D

Michael Raymer, Ph. D

Krishnaprasad Thirunarayan, Ph. D

Andrew Hsu, Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Carral Martínez,David. M.S. Department of Computer Science and Engineering, Wright State University, 2012. Enhancing Description Logics For Rules Coverage.

This thesis is a formal study on how to extend the set of logic constructors of description logics languages, used for knowledge representation in the Semantic Web field to capture some of the previously exclusive rules expressivity.

Description Logics, the logics underpinning the Web Ontology Language OWL, and rules are currently the most prominent paradigms used for modeling knowledge for the Semantic Web. While both of these approaches are based on classical logic, the paradigms also differ significantly, so that naive combinations result in undesirable properties such as undecidability. Recent work has shown that many rules can in fact be expressed in OWL. In this thesis we extend this work to include some types of rules previously excluded. We formally define a set of first order logic rules, C-Rules, which can be expressed within OWL extended with role conjunction. We also show that the use of nominal schemas results in even broader coverage.

After formally defining C-Rules we tried to relax some of the restrictions imposed to roles in the expressive DL language sroiq. As it will be shown in Section **??** the description logic (DL) fragment $\mathcal{ERI}(\sqcap)$, which does not enforce role regularity restrictions, is the core of a larger DL fragment that results in a larger coverage of rules within the DL paradigm. Unfortunately, we show in this thesis that expanding $\mathcal{ERI}(\sqcap)$ with most of the classical DL constructors leads to undecidability. To support this statement, we formally present a set of reductions from the domino problem and the intersection of two free context grammars, both well known undecidable problems, for several minimal DL fragments. These results limit the possible integration of the description logics and rules paradigms and are of significant importance in order to find workable combinations of the two paradigms. Regretfully, most of this results are redundant preventing them for being publishable. Despite this fact we find these undecidability proofs interesting by themselves and include them as part of this thesis.

We also present as part this thesis an addition to OWL 2 syntax to incorporate nominal schemas, which is a new description-logic style extension of OWL 2 which was recently proposed, and which makes is possible to express variable nominal classes within axioms in an OWL 2 ontology. Nominal schemas make it possible to express DL-safe rules of arbitrary arity within the extended OWL

paradigm, hence covering the well-known DL-safe SWRL language. To express this feature, we extend OWL 2 syntax to include necessary and minimal modifications to both Functional and Manchester syntax grammars and mappings from these two syntaxes to Turtle/RDF. We also include several examples to clarify the proposal.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

To Pascal who motivated me through all this work and always tolerates all my distractions with a smile.

This work was partially supported by the National Science Foundation under award 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science

1

Introduction

As already stated in the introduction, the main topic behind this thesis is the extension of the description logics (DLs) expressivity trying to cover some of the previously exclusive roles expressivity. In this first section we give some preliminary information about DL and rules to improve the understanding of the rest of the thesis.

1.1 Description Logics

DLs are a family of knowledge representation languages that serve as the underpinnings for the Web Ontology Language OWL as standardized by the World Wide Web Consortium (W3C). However, DLs have been used in knowledge representation long before the advent of ontological modelling in the context of the Semantic Web, tracing back to rst DL modelling languages in the mid 1980s.

DLs are decidable fragments of first order logic (FOL) with equality are equipped with *formal semantics*: a precise specification of the meaning of DL ontologies. This formal semantics allows humans and machines to precisely understand without ambiguity their intended meanings. Furthermore, these formal semantics allow us to infer additional implicit knowledge from the existing explicit stated facts. We will refer to these computation of new inferences as *reasoning*.

DLs have developed a family of knowledge representation languages ranging from light-weight formalisms for which common inference tasks can be solved in polynomial time to highly expressive logics for which reasoning is intractable. A major design goal for DL is to preserve decidability, at least for the main reasoning tasks such as knowledge base satisfiability.

Theories of a DL are usually referred as *knowledge bases*. DL knowledge bases describe models that are based on individual elements, classes of which elements can be instances, and binary relationships between the elements. Knowledge bases are models based on three kinds of semantic entities: *individuals*, *concepts*, and *roles*. These entities respectively correspond to constants, unary

predicates, and binary predicates in FOL.

In this section we present the basic building blocks allowed in DL. Note that not all the blocks may be allowed in a single DL language. In section 3 we will specify for each profile the allowed constructors for the given fragment.[1]

1.1.0.1 Assertional Facts with ABox axioms

$$Mother(julia) \tag{1.1}$$

$$parentOf(julia,\ john) \tag{1.2}$$

ABox axioms capture knowledge about named individuals. They are equivalent to datalog facts and are included in OWL profile languages considered in this review. We have that axiom 1.1 is a *concept assertion* expressing that individual *julia* is an instance of class *Mother*. Axiom 1.2 is a *role assertion* that connects individual *julia* with *john* by role *parentOf*. Some DL languages support negative assertional axioms such as axioms 1.5 and 1.4.

$$\neg Mother(john) \tag{1.3}$$

$$\neg parentOf(john,\ julia) \tag{1.4}$$

It is important to denote that DLs do not make the *unique name assumption* (UNA), so different names may refer to the same individual unless explicitly state otherwise.

$$julia \not\approx john \tag{1.5}$$

Axiom 1.5 implies that indeed named individuals *julia* and *john* are actually different individuals.

1.1.0.2 Expressing Terminological Knowledge with TBox axioms

$$Mother \sqsubseteq Parent \tag{1.6}$$

$$Person \equiv Human \tag{1.7}$$

TBox (terminological box) axioms allow us to define relations at a terminological level, i.e., relationships between concepts. Axiom 1.6 makes *Mother* a subclass of concept *Parent*, i.e. every individual that belongs to class *Mother* is also in *Parent*. Axiom 1.7 implies equivalence between the concepts *Person* and *Human*. It can be understood as a macro for axioms $Person \sqsubseteq Human$ and $Human \sqsubseteq Person$.

[1] For a more detailed definition of the DL logic constructors check [29].

1.1.0.3 Relationships between Roles with RBox axioms

$$parentOf \sqsubseteq ancestorOf \tag{1.8}$$

$$brotherOf \circ parentOf \sqsubseteq uncleOf \tag{1.9}$$

$$parentOf^{-} \sqsubseteq childrenOf \tag{1.10}$$

$$friendOf \sqcap coworkerOf \sqsubseteq friendAndCoworkerOf \tag{1.11}$$

$$Disjoint(parentOf,\ ChildOf) \tag{1.12}$$

RBox (role box) axioms refer to properties of roles. In a similar way as TBox axioms allow subclass relationships for concepts, RBox axioms support both for *role inclusion* and *role equivalence*, as shown in 1.8. Axiom 1.9 allows us to express *role composition axioms* subsuming several roles in a role chain. Intuitively we have that if *brotherOf(Brad, Chase)* and *parentOf(Chase, John)* are stated explicitly in our knowledge base the reasoning algorithms will be able to infer the new pair *uncleOf(Brad, John)*.

Axiom 1.10 defines a role as the inverse of another role, i.e. if we have *parentOf(julia, john)* and axiom 1.10 as part of the knowledge base then the fact *childrenOf(john, julia)* is entailed. We can express role conjunctions, as shown in axiom 1.11, creating a new relation from two previously existing roles. Axiom 1.12 establishes that two roles are disjoint, and therefore no pair of individuals is connected by these two for any given interpretation of the knowledge base.

1.1.0.4 Boolean Concept Constructors

$$Female \sqcap Parent \tag{1.13}$$

$$Father \sqcup Mother \tag{1.14}$$

$$\neg Female \tag{1.15}$$

With boolean constructors we can construct more complex concepts that can be used combined with TBox axioms. Axiom 1.13 refers to the class of all individuals contained in both the concepts *Female* and *Parent*. The concept defined in axiom 1.14 encompasses all individuals that are contained either in concept *Father* or *Mother*. Class defined in axiom 1.15 refers to all the individuals not contained in class *Female*.

As part of the special DL concepts some DLs languages include concepts top $\top$ and bottom $\bot$. Top concept $\top$ is by default a superclass of all other classes in a DL ontology and bottom $\bot$ refers to the empty class, i.e. the class containing no individuals. These concepts can be respectively translated into FOL by *true* and *false*.

1.1.0.5 Role Restrictions

$$\forall parentOf.Female \tag{1.16}$$

$$\exists parentOf.Person \tag{1.17}$$

$$\exists talksTo.Self \tag{1.18}$$

$$\leq 2 childOf.Parent \tag{1.19}$$

$$\geq 2 childOf.Parent \tag{1.20}$$

These set of logic constructs link both concepts and roles together, including not only the classical existential and universal quantifiers; but also qualified counting quantifiers. Axiom 1.16 refers to the class of all individuals that are only parents of *Female* individuals. The concept described by axiom 1.17 refers to all individuals that are actually in the relation *parentOf* with another individual of the class *Person*. Axiom 1.18 refers to all individuals that are in the relation *talksTo* with themselves. We consider axioms 1.19 and 1.20 self-explanatory after defining the existential and universal quantifiers.

1.1.0.6 Nominals

$$Beatle \equiv \{john\} \sqcup \{paul\} \sqcup \{george\} \sqcup \{ringo\} \tag{1.21}$$

Nominals are concepts with a singleton extension, i.e. exactly one instance. Combining nominals we can express enumerations, such as the one shown in axiom 1.21.

1.1.1 Description Logics to First Order Logic

DL is a decidable fragment of First Order Logic (FOL) with equality and therefore, there exist a transformation from the DL syntactic notation to first order logic. Note that DL makes use of the same model-theoretic semantics as First Order Logic.

We translate some DL axioms into FOL to improve the understandability of the presented constructors in the previous paragraphs as shown in Table 1.1. We have that C, D, and E are concept names and R, T, and S role names.[2] For a full transformation from DL to FOL we point the reader to [4].

[2]Note that a name in an ontology may be either a concept name, a role name, or an individual name, but never both of these at the same time.

Table 1.1: DL to FOL

Description Logics	First Order Logic
$C \sqcap D \sqsubseteq E$	$\forall x(C(x) \wedge D(x) \rightarrow E(x))$
$C \sqsubseteq D \sqcup E$	$\forall x(C(x) \rightarrow D(x) \vee E(x))$
$C \sqsubseteq \exists R.D$	$\forall x(C(x) \rightarrow \exists y\ (R(x,y) \wedge D(y)))$
$C \sqsubseteq \exists R.Self$	$\forall x(C(x) \rightarrow R(x,x))$
$\exists R.C \sqsubseteq D$	$\forall x,y(R(x,y) \wedge C(y) \rightarrow D(x))$
$C \sqsubseteq \forall R.D$	$\forall x,y(C(x) \wedge R(x,y) \rightarrow D(y))$
$\forall R.D \sqsubseteq D$	$\forall x(\forall y(R(x,y) \rightarrow C(y)) \rightarrow D(x))$
$R \sqsubseteq S$	$\forall x,y(R(x,y) \rightarrow S(x,y))$
$R \sqcap S \sqsubseteq T$	$\forall x,y(R(x,y) \wedge S(x,y) \rightarrow T(x,y))$
$R \circ S \sqsubseteq T$	$\forall x,y,z(R(x,y) \wedge S(y,z) \rightarrow T(x,z))$

1.2 Rules

In the broadest sense, a rule could be any statement which says that a certain conclusion must be valid whenever a certain premise is satised, i.e. any statement that could be read as a sentence of the form "if ... then ...". Typical representatives are rules in logic programming or association rules in databases. Rules are logical statements of the form:

$$\bigwedge B_i \rightarrow H$$

Where $\bigwedge B_i$ is called the *body* of the rule and H is the *head*. Whenever the facts in the body of the rule we can derive the fact in the head. Note that a rule of the previous form implies that $\neg H \rightarrow \neg \bigwedge B_i$.

Although both DL and rules paradigms are based on classical logic, they differ significantly and the search for a satisfactory integration is still ongoing [9; **?**]. Both approaches have support different expressivity and none is a superset of the other.

This works focuses in expanding the DL paradigm towards the rules previously exclusive expressivity. Namely, we formally study the rules that can be captured within the DL paradigm by adding role conjunction over complex roles.

2

Extending Description Logic Rules

This chapter extends on the work presented in [23] where it has been shown that, in fact, many rules can be expressed in OWL. We extend this work to include some types of rules previously excluded. We formally define C-Rules, a set of rules that can be embedded directly into OWL extended with role conjunction. We also discuss how our approach can be used in conjunction with previous weaker methods for embedding rules based on nominal schemas.

To express C-Rules in DL notation we employ the DL $\mathcal{SROIQ}(\sqcap_{\exists})$, an extension of $\mathcal{SROIQ}$ [13], which underlies OWL 2 DL. $\mathcal{SROIQ}(\sqcap_{\exists})$ encompasses $\mathcal{SROIQ}$ adding a restricted form of role conjunction.

To introduce our approach, consider the following rule R which cannot be readily expressed in $\mathcal{SROIQ}$ using known techniques. Although R may not have a single directly equivalent axiom in DL, we can transform it into a set of equisatisfiable statements in first-order predicate logic (FOL).

$$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{TaughtByUncle}(x)$$

The example rule R can be represented as an equisatisfiable set of statements:

- $\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \rightarrow \text{hasUncle}(x, z)$
- $\text{hasUncle}(x, z) \wedge \text{hasTeacher}(x, z) \rightarrow \text{hasTeacherAndUncle}(x, z)$
- $\text{hasTeacherAndUncle}(x, z) \rightarrow \text{TaughtByUncle}(x)$

This set of FOL statements can then be translated into

- $\text{hasFather} \circ \text{hasBrother} \sqsubseteq \text{hasUncle}$
- $\text{hasUncle} \sqcap \text{hasTeacher} \sqsubseteq \text{hasTeacherAndUncle}$
- $\exists \text{hasTeacherAndUncle}.\top \sqsubseteq \text{TaughtByUncle}$

and therefore the rule R can be expressed in DL notation.

Although some rules fall under our definition and are expressible using these combinations of role constructors, there are even more complex rules that cannot be simplified using the approach presented in this paper.

A prominently discussed idea for retaining decidability, and still be able to express complex rules, is to restrict the applicability of rules to named individuals. Rules with this kind of semantics are called DL-safe, and the combination of OWL and DL-safe rules is indeed decidable [14; 30]. In this paper we also discuss the use of nominal schemas to express complex rules. Nominal schemas are a DL constructor presented in [26] described as "variable nominal classes." Restricted to stand only for named individuals as DL-safe variables, nominal schemas have the advantage of allowing us to express complex rules in native OWL notation.

The plan for the rest of the chapter is as follows. After providing some preliminaries in Section 2.1, the chapter continues in Section 2.2 with the formal definition of C-Rules with unary predicates in the head. Section 2.3 extends the approach presented in the previous section to rules with binary predicates in the head. Section 4.4 contains some examples. Section 2.5 contains the discussion about rules and nominal schemas. Section 2.6 includes the conclusions of the chapter and future work.

2.1 Preliminaries

We introduce $\mathcal{SROIQ}(\sqcap_{\exists})$, a DL fragment that adds role conjunction, in a restricted way, to $\mathcal{SROIQ}$ [13]. Axioms of the form $R_1 \sqcap R_2 \sqsubseteq V$ are allowed in $\mathcal{SROIQ}(\sqcap_{\exists})$, where R_1 and R_2 are two (possibly complex) roles.[1]

Roles which appear on the right hand sides of axioms of the form $R_1 \sqcap R_2 \sqsubseteq V$ are restricted to only appear in concepts of the form $\exists V.C$. Although this precondition might look very restrictive, it suffices for the use of role conjunction for expressing rules, as discussed in this chapter. As a technical note, in terms of regularity of RBoxes (required for decidability), we assume that for a role V appearing in an axiom $R_1 \sqcap R_2 \sqsubseteq V$ we have that both $R_1 \prec V$ and $R_2 \prec V$ ($\prec$ indicates the order in a regular role hierarchy).

$\mathcal{SROIQ}(\sqcap_{\exists})$ bears the same semantics as $\mathcal{SROIQ}$, with the exception of the role conjunction constructor. The formal semantics is as usual (see, e.g., [36]), and for lack of space we do not repeat it here. Note that it follows easily from the arguments laid out in [36] that $\mathcal{SROIQ}(\sqcap_{\exists})$ is decidable.

[1] In a sense, role conjunction was already implicit in [24], and is also used in [23], for a similar purpose.

2.1.1 Description Logic Rules and Graph Notation

We define *U-Rules* (respectively, *B-Rules*) as rules of the form $\bigwedge B_i \rightarrow H$, where all B_i are unary or binary predicates and H is a unary (binary) predicate.[2] We refer to $\bigwedge B_i$ and H as the *body* and the *head* of the rule respectively. We assume without loss of generality that at least one of the variables in the head of the rule also appears in the body of the rule at least once.[3]

Let R be any given U-Rule or B-Rule. We can define an *undirected graph* G_R derived from R as a set of vertices and edges s.t. every variable in the body of R is a vertex of G_R and G_R contains an edge (t, u) if $S(t, u)$ is a binary predicate in the body of R, and t and u are different variables. Note, that rules containing a predicate of the form $R(x, x)$ where R is a complex role cannot be expressed in $\mathcal{SROIQ}$, as it will be shown in Section 2.2.1. Consequently, if this is the case the reduction process cannot be performed. Graphs will be used across the chapter to represent and reduce rules in an easy and intuitive way.

Note that constants, and even binary predicates containing only one variable, are not included in the graph. Having a prefixed meaning, constants can be simplified independently and therefore, there is no need to relate them to other elements in the graph. On the other hand, FOL variables, having shared non-fixed meanings, need to keep the links that determine their relation to the predicates in the rule where they appear.

We have defined undirected graphs G_R as sets and consequently there cannot be repetitions amongst its elements. Even if two different binary predicates relate the same pair of terms, in the graph these predicates map to the same single edge. Note also that we work with undirected graphs and therefore, elements (u, t) and (t, u) stand for different representations of the same edge.

For an undirected graph G_R, derived from a U-Rule or a B-rule R, we define a vertex as a *root vertex* r_R if it has been derived from a variable appearing both in the head and the body of the rule. Note that by our standing assumption, there will always be at least one root vertex for any given graph G_R.

We define, for two given vertices t and u, to be *directly connected* if $(t, u) \in G_R$. We define, for two given vertices t and u, to be *connected* as the symmetric transitive closure of being directly connected. We assume without loss of generality that any two vertices in the graph are connected.[4] We define the *degree of a vertex* $d(t)$ as the number of different edges (t, u) in the graph G_R where

[2] In our context unary predicates will refer to any kind of valid unary $\mathcal{SROIQ}(\sqcap_{\exists})$ concept in negation normal form, either in the head or in the body of the rule.

[3] Note that, if none of the variables in the head of the rule appears in the body, we can always include one of them in the body using a unary top predicate.

[4] Again, we can connect any two variables in a rule using the universal role—although regularity issues need to be taken into consideration here.

t appears.

2.2 Description Logic Rules in $\mathcal{SROIQ}(\sqcap_\exists)$

In this section, we formally describe rules that can be expressed in the DL fragment $\mathcal{SROIQ}(\sqcap_\exists)$. While close in general spirit to the brief exhibit in [25, Section 8.3], our treatment is quite different.

Definition We propose in this definition a formal method to verify if a U-Rule R is expressible in $\mathcal{SROIQ}(\sqcap_\exists)$. Given a graph G_R derived from a given rule R, repeat non-deterministically and exhaustively:

- Substitute a pair of edges (t, u) and (u, v) by a single edge (t, v) and eliminate vertex u if $d(u) = 2$ and u is a non-root vertex.

$$G_R := \{G_R \setminus \{(t, u),\, (u, v),\, u\} |\, d(u) = 2,\, u \neq r_R\} \cup \{(t, v)\}$$

- Eliminate an edge (t, u) and a vertex u where $d(u) = 1$ and u is a non-root vertex.

$$G_R := \{G_R \setminus \{(t, u),\, u\} \,|(t, u) \in G_R,\, d(u) = 1,\, u \neq r_R\}$$

A rule R is expressible in $\mathcal{SROIQ}(\sqcap_\exists)$ if the graph G_R gets reduced to the root vertex after the reduction process.

The reduction process defined in Definition 2.2 only halts under two different situations. Either the graph G_R gets reduced to a single vertex or, for every non-root vertex u in the graph we have that $d(u) \geq 3$ (possibly after several reduction steps). Steps 1 and 2 presented in the previous reduction process can remove a vertex u if $d(u) = 2$ or $d(u) = 1$ respectively if u is a non-root vertex. It is obvious that a graph containing a non-root vertex u s.t. $d(u) < 3$ can be reduced at least one step further, and a graph where $d(u) \geq 3$ for every non-root vertex $u \in G_R$ cannot be simplified.

The process presented in Definition 2.2 presents a formal approach to determine if a given rule R is expressible in $\mathcal{SROIQ}(\sqcap_\exists)$. In the sequel we explain how, from this graph reduction approach, we can derive a set of $\mathcal{SROIQ}(\sqcap_\exists)$ axioms equivalent to the original rule R. Formally stated, the following two lemmas and theorem hold. Their correctness is shown in Section 2.2.1.

Note that, even if we do not provide an explicit definition for rules that are not expressible in $\mathcal{SROIQ}(\sqcap_\exists)$ this can be easily inferred. Rules with four nodes that are fully connected through paths not containing any of those nodes are not expressible. Even if all the other nodes in the paths get simplified these four nodes will form a clique where each one of them has degree three or higher. Therefore, they cannot be reduced using the approach presented in this chapter.

Lemma 2.2.1. *Let R be a U-Rule with a derived graph G_R. Every transformation performed on G_R as explained in Definition 2.2 leads to a new reduced graph $G_{R'}$. Then a new rule R_1 can be constructed from R s.t. we can derive a graph G_{R_1} from R_1 and $G_{R_1} = G_{R'}$. Furthermore, there exist a set of $\mathcal{SROIQ}(\sqcap_\exists)$ statements α_1 s.t. $\{R_1\} \cup \alpha_1 \equiv \{R\}$.*

Definition The process defined in Lemma 2.2.1 can be repeated iteratively producing a new rule from every rule derived from R s.t. $R \equiv R_1 \cup \alpha_1$, $R_1 \equiv R_2 \cup \alpha_2, ..., R_{t-1} \equiv R_t \cup \alpha_t$. R_t is a rule with a derived graph G_{R_t} s.t. G_{R_t} cannot be reduced anymore. If G_{R_t} has been reduced to a single vertex we call R_t a *terminal rule.*

Lemma 2.2.2. *A terminal rule R_t is directly expressible as a $\mathcal{SROIQ}(\sqcap_\exists)$ axiom.*

Intuitively, the simplifications done on the graph will be mirrored in the rule. Through this iterative process we can start reducing the rule, splitting it into several $\mathcal{SROIQ}$ axioms that preserve equisatisfiability. When the rule gets reduced to a terminal rule it can be directly translated into $\mathcal{SROIQ}(\sqcap_\exists)$. Adding this translation to the set of previous axioms derived in the reduction process we obtain an equivalent set of atoms in $\mathcal{SROIQ}(\sqcap_\exists)$. The following Theorem follows directly from Lemmas 2.2.1 and 2.2.2.

Theorem 2.2.3. *Assume that a rule R is reduced to terminal rule R_t. Then the original knowledge base KB containing R is equisatisfiable w.r.t. a new knowledge base KB'. KB' is obtained from KB by substituting R by $\alpha_1 \cup ... \cup \alpha_{t-1} \cup \{R_t\}$ where R_t is the direct translation of the terminal rule R_t and $\alpha_1, ..., \alpha_{t-1}$ are the sets of axioms produced in every step during the reduction process of rule R.*

2.2.1 Satisifiability Preserving Transformations

We show how to carry out the mentioned graph transformations for a given rule R, preserving equisatisfiability in every step. For every transformation we present a table with three columns. The first column shows the initial subset of R that will be replaced in the next iteration of the reduction process. The second column includes the new simplified subset. By substituting the initial subset by the simplified one in the original rule R we obtain the new simplified rule. Column three shows all the $\mathcal{SROIQ}(\sqcap_\exists)$ statements that we need to add to the knowledge base in order to preserve equivalence.

Roling Up Unary Predicates We start by proving that unary predicates can be automatically embedded into binary predicates using role constructors. Therefore, these predicates do not need

to be considered when we build the graph to know if a description logic rule R is expressible in $\mathcal{SROIQ}(\sqcap_\exists)$. This technique, called rolification, is presented in [**?**].

Initial rule subset	Eq Subset	Set α
$V(x,y) \wedge D(y)$	$S(x,y)$	$D \sqsubseteq \exists W.\text{Self}$
		$V \circ W \sqsubseteq S$

Hereby, W and S are fresh names not already appearing in the knowledge base.

Proposition 2.2.4. *Let KB be a knowledge base containing a U-Rule R s.t. $V(x,y) \wedge D(y)$ appears in R. From KB we can construct an equivalent knowledge base KB' s.t. $KB' := \{KB \backslash R\} \cup \{R'\} \cup \alpha$, where R' is a new U-Rule obtained from R by substituting $V(x,y) \wedge D(y)$ with $S(x,y)$ and the set α contains the axioms $D \equiv \exists W.Self$ and $V \circ W \sqsubseteq S$, with W and S fresh predicate names. We prove that KB and KB' are equisatisfiable knowledge bases.*

Proof. Assume M is a model for the KB. Obviously M models R. From M we can build a model M' for KB' s.t. M' is identical to M except for the mappings $W^{M'} = \{\ (b,b) \mid b \in D^M\ \}$ and $S^{M'} = \{\ (a,b) \mid (a,b) \in V^M$ and $(b,b) \in W^{M'}\ \}$. These mappings are enforced by the new α axioms added to the knowledge base. Obviously, if M models R, M' models axioms $\{R'\} \cup \alpha$ and hence, M' is a model for KB'.

Now assume that M' is a model of $\{R'\} \cup \alpha$. Then we have that $M = M'$ is a model for R. Consider ground instances of the rule and assume that $M \models \bigwedge B_i$. Now we need to prove that $M \models H$. Since we assume that $M \models \bigwedge B_i$ we have that M models every B_i in the body of the rule, in particular $M \models V(a,b) \wedge D(b)$. Hence, $(b,b) \in W^{M'}$ and $(a,b) \in S^{M'}$. Then we have that $M \models \bigwedge B'_i$ where B'_i is the R' body. Therefore $M \models H'$, where H' is the head of the new rule R'. We have that $H' = H$, hence, we have shown that that $M \models H$ being H. Consequently M models R and is a model for KB. □ □

All the other transformations presented below can be proved in a similar way. We thus refrain from including any more detailed formal arguments for them.

Note that $V(x,y) \wedge D(x)$ can be simplified in a similar way. In the following, W and S are fresh role names in the ontology.

Initial rule subset	Eq Subset	Set α
$V(x,y) \wedge D(x)$	$S(x,y)$	$D \sqsubseteq \exists W.\text{Self}$
		$W \circ V \sqsubseteq S$

Undirected Graph The direction of the edges in the graph can be changed using the inverse role constructor. Therefore, there is no need for our algorithm in Definition 2.2 to check the direction of binary predicates when we apply different simplifications. Below, S is fresh role name in the knowledge base.

Rule Subset	Eq Subset	Set α to the KB
$R(x,y)$	$S(y,x)$	$R^- \sqsubseteq S$

Reducing Vertices of Degree Two

Rule Subset	Eq Subset	Set α to the KB
$R(x,y) \wedge W(y,z)$	$S(x,z)$	$R \circ W \sqsubseteq S$

Hereby, $d(y) = 2$ and S is fresh role name in the knowledge base.

Note that when we simplify a vertex by using a role chain we loose the reference to the term u in the middle of the role chain for a given rule R. Therefore this can only be executed for terms u with a degree of two, without further references in other predicates of the rule R.

Reducing Vertices of Degree One

Rule Subset	Eq Subset	Set α to the KB
$R(x,y) \wedge V(y,z)$	$R(x,y) \wedge D(y)$	$\exists V.\top \sqsubseteq D$

Hereby, $d(z) = 1$ and D is a fresh concept name in the knowledge base.

Note that even if we part from a very similar subset rule as in the previous subsection we follow a different method here. The fresh unary predicate produced can be simplified as shown in earlier sections.

Simplifying Binary Predicates with One Constant

Initial rule subset	Eq Subset	Set α
$V(x,a)$	$S(x)$	$\exists V.\{a\} \sqsubseteq S$

Hereby, S is a fresh unary predicate and a is a is a constant.

The fresh unary predicate can be simplified as shown in earlier sections. Note that we can always swap the order of the terms in a predicate of the form $V(a,x)$.

Simplifying Binary Predicates of the Form (x,x)

Initial rule subset	Eq Subset	Set α
$V(x,x)$	$S(x)$	$\exists V.\text{Self} \sqsubseteq S$

Hereby, S is a fresh unary predicate.

Using this simplification whenever possible, there is no need to consider these kind of predicates in the graph. Note that this can only be done if V is a simple role w.r.t. role box in the knowledge base. To retain decidability $\mathcal{SROIQ}$ does not allow to use complex in a concept of the form $\exists C$.Self.

Unifying Binary Predicates

Rule Subset	Eq Subset	Set α to the KB
$R(x,y) \wedge V(x,y)$	$S(x,y)$	$R \sqcap V \sqsubseteq S$

Hereby, S is a fresh role name in the knowledge base.

Any pair of binary predicates can be unified if both contain the same pair of variables. Note that, even if the variables do not appear in the same order, we can use the inverse role construct to align them. We assume without loss of generality that every pair of binary predicates containing the same pair of variables in a rule is automatically unified and therefore, we can define graphs as sets without repetitions of the same edge. Note that the unification of binary predicates needs to be done with a higher priority than other transformations, such as the reduction of vertices of degree two. If not cycles may be reduced to predicates of the form $R(x,x)$ where R is a complex role that cannot be simplified using the $\exists R$.Self.

The transformations just shown correspond to graph transformations as mentioned in Definition 2.2. The arguments just given thus constitute a proof of Lemma 2.2.1. Note that the order of the transformations is non-deterministic. This is a kind of "don't care" determinism, where the order in which we apply the rules does not matter. We further elaborate about this in Section 4.4.

Translating Terminal Rules A terminal rule R is a rule of the form $\bigwedge B_i \rightarrow H$. We have that the body of the rule $\bigwedge B_i$ contains one, and at most one, free FOL variable x appearing only once in a unary predicate of the form $B(x)$ (the graph has been reduced to the root vertex, and therefore, there is only one variable left appearing only once). The body $\bigwedge B_i$ might also contain other predicates of the form $C(a)$ or $R(b,c)$ s.t. a, b, and c are constants. The head H is composed of a single unary predicate $H(x)$ s.t. x is the same free variable that appears in the body.

A terminal rule R is translated into a DL inclusion axiom of the form $\bigsqcap B_i \sqsubseteq H$. This axiom contains a fresh concept H on the right hand side of the role inclusion axiom and a concept intersection on the left hand side featuring the next elements:

- A fresh concept B standing for the unary predicate $B(x)$ s.t. x is the only free variable appearing in the rule.

- A concept $\exists U.(C \sqcap \{a\})$ for every unary predicate of the form $C(a)$ appearing in the body where a is a constant.
- A concept $\exists U.(\{b\} \sqcap \exists R.\{c\})$ for every binary predicate of the form $R(b, c)$ appearing in the body of the rule where b and c are constants.

The argument just given also constitutes a proof of Lemma 2.2.2.

2.3 Rules with Binary Predicates in the Head

We can extend our approach to cover rules with binary predicates in the head. As already stated, at least one variable in the head of the rule needs to appear in the body. Attending to this fact we need to consider two different situations.

If only one of the terms in the head of the rule appears in the body the simplification is straightforward. We just need to substitute the binary predicate $R(x, y)$[5] in the head by a fresh unary predicate $C(x)$ and add the axiom $\exists R.\top \sqsubseteq C$ to the $\mathcal{SROIQ}(\sqcap_{\exists})$ knowledge base. After this modification the rule can be reduced using the same approach presented in the previous section.

If both variables appearing in the head of the rule appear in the body we need to slightly modify our method presented in Section 2.2. In this case, we consider both variables in the head as root vertices. Now, if the rule is expressible in $\mathcal{SROIQ}(\sqcap_{\exists})$ the graph gets reduced to a single edge containing both variables in the head. This new kind of terminal rule can be expressed in $\mathcal{SROIQ}(\sqcap_{\exists})$ using a role inclusion axiom $S \sqsubseteq R$.

Theorem 2.3.1. *Let R be a B-Rule s.t. $S(x, y)$ is the predicate in the head H of R and both x and y appear in the body $\bigwedge B_i$ of the rule.*

Given a graph G_R derived from rule R, where we consider both x and y to be root vertices. Then exhaustively apply steps 1 and 2 from Definition 2.2.

If after the reduction process, the graph G_R gets reduced to a single edge, then rule R is expressible as a $\mathcal{SROIQ}(\sqcap_{\exists})$axiom.

Again, we see that any G_R where every vertex u has $d(u) \geq 3$ (possibly obtained after several reduction steps) cannot be simplified. Otherwise the graph can be reduced to a single edge (u, t) s.t. $u \in H$ and $t \in H$ with H the head of the rule. Note that the procedure is almost the same as in Section 2.2, except for the accepting condition.

[5] Assume x is the root vertex. Otherwise use the inverse role constructor to change the order of the terms in the predicate.

The process to translate the rule into a set of equivalent $\mathcal{SROIQ}(\sqcap_\exists)$ statements and proofs remain the same as the one presented in Section 2.2 except for the trivial translation of the terminal rule.

It is important to remark that in some cases a B-Rule may not be expressible while a U-Rule with the same body is. The second vertex might block a possible role reduction forbidding further simplifications. As an example we have that $R_1(x,y) \wedge R_2(x,w) \wedge R_3(w,y) \wedge R_4(y,z) \wedge R_5(w,z) \rightarrow C(x)$ is expressible in $\mathcal{SROIQ}(\sqcap_\exists)$ using our approach, while $R_1(x,y) \wedge R_2(x,w) \wedge R_3(w,y) \wedge R_4(y,z) \wedge R_5(w,z) \rightarrow C(x,z)$ is not.

Definition Formally, by *C-Rules* we mean the collection of all rules which are U-Rules or B-Rules and which are expressible in $\mathcal{SROIQ}$ using the approach presented in this chapter.

2.4 Rule Translation Examples

We start with a worked example for our transformation. As initial rule, we use

$$A(x,y) \wedge B(y) \wedge C(z,y) \wedge D(y,z) \wedge E(x,a) \wedge F(x,z) \wedge Y(a,b) \rightarrow Z(x)$$

where a and b are constant and x, y and z are free variables. Transformations following the discussion from Section 2.2 are detailed in Table 2.1.

Note that the rule listed in step 6 of Table 2.1 can already be directly translated to $\mathcal{SROIQ}(\sqcap_\exists)$ as $\exists M.\top \sqcap \exists E.\{a\} \sqcap \exists U.(\{a\} \sqcap \exists Y.\{b\}) \sqsubseteq Z$. But to improve readability of the chapter, our rule reduction approach has been presented in a simpler form, avoiding such shortcuts. So, although the method shown is sound and correct, there are U-Rules and B-Rules, as the one presented in the example, where at some step of the reduction process no further simplifications are strictly required. An earlier translation of the rule reduces the number of statements that need to be added to the knowledge base. Recall, in particular, that rules with tree shaped graphs are directly expressible in DL [**?**; 23].

Also, we have that reductions according to our transformations are applied non-deterministically. Although any rule reduction leading to a terminal rule is essentially correct, there might be differences in the set of axioms added to the knowledge base. For example, let R be a U-Rule containing the binary predicates $A(x,y)$ and $B(y,z)$ s.t. both y and z are variables not appearing anywhere else in the rule (hence, we have that $d(y) = 2$ and $d(z) = 1$). In the next reduction step, we can decide which variable, y or z, we want to erase.

Assuming we want to reduce $A(x,y)$ and $B(y,z)$ to $Z(x)$, there are two different ways of doing so, namely (1) first reducing y, and (2) first reducing z. In the first case, we end up with two

Table 2.1: Reduction example. For every step substitute the rule in the previous row by the one in the current one and add the axioms on the second column to the knowledge base.

Step	Add to KB	Rule
1. Original Rule		$A(x,y) \wedge B(y) \wedge C(z,y) \wedge D(y,z) \wedge$ $E(x,a) \wedge F(x,z) \wedge Y(a,b) \rightarrow Z(x)$
2. Invert C	$C^- \sqsubseteq H$	$A(x,y) \wedge B(y) \wedge H(y,z) \wedge D(y,z) \wedge$ $E(x,a) \wedge F(x,z) \wedge Y(a,b) \rightarrow Z(x)$
3. Intersect D and H	$D \sqcap H \sqsubseteq I$	$A(x,y) \wedge B(y) \wedge I(y,z) \wedge E(x,a) \wedge$ $F(x,z) \wedge Y(a,b) \rightarrow Z(x)$
4. Role Up B	$B \sqsubseteq \exists J.\text{Self}$ $A \circ J \sqsubseteq K$	$K(x,y) \wedge I(y,z) \wedge E(x,a) \wedge$ $F(x,z) \wedge Y(a,b) \rightarrow Z(x)$
5. Simplify E	$\exists E.\{a\} \sqsubseteq L$	$K(x,y) \wedge I(y,z) \wedge L(x) \wedge$ $F(x,z) \wedge Y(a,b) \rightarrow Z(x)$
6. Role Up L	$L \sqsubseteq \exists M.\text{Self}$ $M \circ F \sqsubseteq N$	$K(x,y) \wedge I(y,z) \wedge$ $N(x,z) \wedge Y(a,b) \rightarrow Z(x)$
7. Reduce y	$K \circ I \sqsubseteq O$	$O(x,z) \wedge N(x,z) \wedge Y(a,b) \rightarrow Z(x)$
8. Intersect N and O	$N \sqcap O \sqsubseteq P$	$P(x,z) \wedge Y(a,b) \rightarrow Z(x)$
9. Reduce z	$\exists P.\top \sqsubseteq Q$	$Q(x) \wedge Y(a,b) \rightarrow Z(x)$
10. Translate Terminal Rule		$Q \sqcap \exists U.(\{a\} \sqcap \exists Y.\{b\}) \sqsubseteq Z$

axioms $A \circ B \sqsubseteq C$ and $\exists C.\top \sqsubseteq Z$, while in the second case four axioms are required $\exists B.\top \sqsubseteq D$, $D \sqsubseteq \exists E.\text{Self}$, $A \circ E \sqsubseteq F$, and $\exists F.\top \sqsubseteq Z$. Note that giving priority to the reduction of variables with degree 2 reduces the number of required axioms to preserve equisatisfiability.

The regularity issue. In order to preserve decidability, $\mathcal{SROIQ}(\sqcap_\exists)$ enforces a strict partial order on complex roles (known as the *regularity* condition). When a C-Rule R is translated into $\mathcal{SROIQ}$, we add many new complex role inclusions axioms to the Rbox. These new roles may violate the partial order established by previous roles or even contradict role regularity by themselves. After the reduction of a C-Rule and the inclusion of new produced $\mathcal{SROIQ}(\sqcap_\exists)$ axioms, role regularity needs to be carefully checked in order to preserve decidability.

It is important to note that only the translation of expressible B-Rules might cause these role regularity violations. Although the role regularity hierarchy is modified in many steps of our rule reduction approach note that for every statement $R \prec S$ added we have that S is a fresh role.

Fresh roles do not appear in any other part of the role hierarchy and therefore they cannot produce violations of the irreflexive order.

The only step of the process where the RBox may loose its regularity is in the translation of a terminal B-Rule. Adding this last axiom of the form $R \sqsubseteq S$ also adds the statement $R \prec S$ to the role hierarchy where S is a role which may have appear in any other part of the knowledge base.

Let us look at an example of a knowledge base where the reduction of some of the rules leads to role regularity violations. Let KB be a knowledge base containing the following rule.

$$\text{TeacherOf}(y, x) \wedge \text{ReviewerOf}(y, z) \wedge \text{AuthorOf}(x, z) \rightarrow \text{IllegalReviewerOf}(y, z)$$

This rule places a pair of individuals under the binary predicate IllegalReviewerOf if the first is a teacher of the student who is the author of the reviewed chapter. It can be transformed into the following set of $\mathcal{SROIQ}(\sqcap_\exists)$ axioms.

$$\begin{aligned} \text{TeacherOf} \circ \text{AuthorOf} &\sqsubseteq R_1 \\ \text{ReviewerOf} \sqcap R_1 &\sqsubseteq R_2 \\ R_2 &\sqsubseteq \text{IllegalReviewer} \end{aligned}$$

From these axioms, we obtain the relations TeacherOf $\prec R_1$, AuthorOf $\prec R_1$, ReviewerOf $\prec R_2$, $R_1 \prec R_2$, and $R_2 \prec$ IllegalReviewer, which entail the statement ReviewerOf $\prec$ IllegalReviewerOf. If would be natural to also add the axiom IllegalReviewerOf $\sqsubseteq$ ReviewerOf to the same ontology. However, the inclusion of this axiom adds the statement IllegalReviewerOf $\prec$ ReviewerOf which then violates role regularity.

A workaround to this issue, however, is possible, namely by employing nominal schemas, and we will return to this issue at the end of the next section.

2.5 Using Nominal Schemas and $\mathcal{SROIQV}(\sqcap_\exists)$

In earlier sections of this chapter we have shown how to translate some FOL rules into DL notation. Although some rules can be translated to $\mathcal{SROIQ}(\sqcap_\exists)$ using the presented approach there are still more complex rules that cannot be simplified in the same way. To express these rules we employ the DL $\mathcal{SROIQV}(\sqcap_\exists)$.

$\mathcal{SROIQV}(\sqcap_\exists)$ adds nominal schemas, a DL constructor that can be used as "variable nominal classes," to the previously described $\mathcal{SROIQ}(\sqcap_\exists)$. We will refrain from introducing all formal details and refer the reader to [18; 20; 21; 26] for this. While the semantic intuition behind nominal schemas is the same as that behind DL-safe variables, nominal schemas integrate seamlessly with DL syntax.

As a consequence, the DL fragment $\mathcal{SROIQV}(\sqcap_\exists)$ encompasses DL-safe variables while staying within the DL/OWL language paradigm avoiding the use of hybrid approaches.

Using these nominal schemas we are able to express FOL rules that are not part of the treatment in Sections 2.2 and 2.3. Consider, for example, the rule

$$R_1(x,y) \wedge R_2(x,z) \wedge R_3(x,w) \wedge R_4(y,z) \wedge R_5(y,w) \wedge R_6(w,z) \rightarrow C(x). \tag{2.1}$$

Using $\{z\}$ and $\{w\}$ as nominal schemas, we can express it as

$$\exists R_1.(\exists R_4.\{z\} \sqcap \exists R_5.\{w\}) \sqcap \exists R_2.\{z\} \sqcap \exists R_3.(\{w\} \sqcap \exists.R_6.\{z\}) \sqsubseteq C$$

Note that, as already stated, nominal schemas do not share the same semantics defined for FOL variables. Nominal schemas, as DL-safe variables, are restricted to stand only for nominals which are explicitly present in the knowledge base, while FOL variables can represent both named and unknown individuals. Therefore, the statements presented in the example just given are not strictly equivalent. Despite this fact, nominal schemas allow us to retain most of the entailments from the original FOL axiom without increasing the worst-case complexity of the DL fragment.

Although nominal schemas do not increase the worst-case complexity of the language [26], the number of different nominal schemas per axiom can affect the performance of the reasoning process [7; 20]. It is therefore desirable to use as few nominal schemas as possible.

We now discuss two different ways of translating complex rules into DLs. First we prove the following.

Theorem 2.5.1. *Any U-Rule or B-Rule R containing m different free variables, where $m > 3$, can be directly expressed in DL using n nominal schemas s.t. $n \leq m - 2$.*

Proof. Given a rule R, firstly role up to simplify all binary predicates containing one constant as shown in Section 2.2. All binary predicates in the rule containing the same pair of variables are also replaced by a single binary predicate as described under *Unifying Binary Predicates* in Section 2.2.

Due to these transformations, we can now assume without loss of generality that the rule R contains only unary predicates with a constant, binary predicates with two constants, and binary predicates with two variables as arguments.

Now choose two variables x and y s.t. x is a root vertex and y is not. Using the inverse role construct we can now swap arguments in binary predicates s.t. x is always appearing in the first argument and y is in the first argument of every predicate where the other variable is not x. The variables selected will be the only ones not substituted by a nominal schema in the translated rule.

The rule body is now translated as shown in Table 3.5. The resulting DL expressions are joined by conjunction. $\bigwedge B_i(y)$, $R(x,y)$, and $\bigwedge R_i(y,v_i)$ are all the predicates where y appears.

Table 2.2: Translating Rules with Nominal Schemas

Predicate type	FOL	DL
Unary Predicates with 1 constant	$B(a)$	$\exists U.(\{a\} \sqcap B)$
Binary Predicates with 2 constants	$R(a, b)$	$\exists U.(\{a\} \sqcap \exists R.\{b\})$
Binary Predicates containing x and not y	$R(x, v)$	$\exists R.\{v\}$
Unary Predicates containing x	$B(x)$	B
Binary and Unary Predicates containing y	$R(x, y) \wedge \bigwedge B_i(y) \wedge \bigwedge R_i(y, v_i)$	$\exists R.(\bigsqcap B_i \sqcap \bigsqcap R_i.\{v_i\})$
Unary Predicates not containing x, y, or constants	$B(v)$	$\exists U.(\{v\} \sqcap B)$
Binary Predicates not containing x, y, or constants	$R(v, u)$	$\exists U.(\{v\} \sqcap \exists R.\{u\})$

Finally, the head $H(x)$ can be rewritten into the concept H (or if it is a binary predicate $H(x, z)$, a concept of the form $\exists H.\{z\}$), and the implication arrow replaced by class inclusion $\sqsubseteq$.

It is straightforward to formally verify the correctness of this transformation, and parts of the proof are simliar to the correctness proof from [26] for the embedding of binary Datalog into $\mathcal{SROIQV}$.

Clearly, the number of nominal schemas used to represent rule R is $n - 2$, the total number of free variables minus 2. □ □

With the transformation just given, rule (2.1) can be rewritten as

$$\exists R_1.(\exists R_4.\{z\} \sqcap \exists R_5.\{w\}) \sqcap \exists R_2.\{z\} \sqcap \exists R_3.\{w\} \sqcap \exists.U(\{w\} \sqcap \exists.R_6.\{z\}) \sqsubseteq C$$

As another example, the following rule transforms into the subsequent axiom.

$$R_1(x, y) \wedge R_2(y, z) \wedge R_3(w, z) \wedge R_4(x, z) \wedge R_5(y, w) \wedge R_6(w, u) \wedge R_7(y, u) \rightarrow H(x, u)$$

$$\exists R_1.(\exists R_2.\{z\} \sqcap \exists R_5.\{w\} \sqcap \exists R_7.\{u\}) \sqcap \exists U.(\exists\{w\} \sqcap \exists R_4.\{z\}) \sqcap \exists R_4.\{z\} \sqcap \exists U.(\{w\} \sqcap \exists R_6.\{u\}) \sqsubseteq \exists H.\{u\}$$

Theorem 2.5.2. *Any U-Rule or B-Rule R containing m different free variables can be expressed in DL by fully grounding $m - 3$ free variables in R.*

Proof. By grounding every variable but three in the rule to named individuals we end up with a larger number of rules s.t. each one of them contains only three different free variables.[6] All these new grounded rules are expressible in DL using the approach presented in Section 3 of this chapter. □ □

While the first of the approaches just mentioned allows us to represent all knowledge in $\mathcal{SROIQV}(\sqcap_\exists)$, the second one, although initially looking more efficient, requires preprocessing steps. Further research and algorithms are required to smartly deal with nominal schemas other than through such grounding, a cumbersome technique that requires too much space and time for current reasoners [7; 20].

Let us finally return to the regularity issue discussed at the very end of Section 4.4. In the example discussed there, if we desire to also add the statement IllegalReviewerOf $\sqsubseteq$ ReviewerOf to the knowledge base, we cannot do so directly without violating regularity. Using nominal schemas, however, we can weaken this axiom to the form

$$\exists \text{IllegalReviewerOf}.\{x\} \sqsubseteq \exists \text{ReviewerOf}.\{x\}$$

(or, e.g., to

$$\exists \text{IllegalReviewerOf}^-.\{x\} \sqsubseteq \exists \text{ReviewerOf}^-.\{x\}$$

or to both), where $\{x\}$ is a nominal schema. Essentially, this means that the role inclusion will apply in case the first argument or the second argument (the filler) is a known individual. I.e., the individuals connected by the IllegalReviewerOf property are not both are unnamed. While this is weaker than the standard semantics, it should provide a viable workaround in many cases. Also note that, alternatively, the regularity violation could be avoided by using a similarly weakened form of any of the other statements involved in the violation.

2.6 Conclusions

This chapter presents an extension of previous work on Description Logic Rules. We specify a translation of rules into OWL extended by role conjunction (more precisely, the description logic $\mathcal{SROIQ}(\sqcap_\exists)$), which strengthens previously obtained such translations. In essense, our work shows that the fragment of Datalog which can be expressed in description logics is larger than previously assumed.

[6] Note that any rule with three variables can be reduced used our approach. Having only three nodes in the graph for all of them we have that $d(u) \leq 2$ and therefore all of them can be reduced.

We furthermore included a discussion proposing two approaches to express more complex rules within the DL notation. Two different options are considered, the use of nominal schemas and fully grounding of some variables to named individuals. While the former might present a higher complexity, it allows to express these rules in native DL/OWL notation and avoid some cumbersome preprocessing steps.

Future work includes the development of a goal directed algorithm that can solve inference problems in $\mathcal{SROIQV}(\sqcap_\exists)$ possibly including some other role constructs (probably the extension of a current tableau algorithm). This algorithm could serve as basis for practical implementations of reasoners that include role constructs amongst their features.

Also, the development of APIs that can automatically validate FOL rules as DL expressible and translate them into sets of equisatisfiable OWL axioms might be a very useful tool. Although some aspects of modeling ontologies, such as building concept hierarchies, are very intuitive when we work with DL/OWL languages, the translation of DL rules, as shown in this chapter, may not be so straightforward for users that do not have a strong background in more formal logics. Additional tool support will be required to provide convenient modeling interfaces.

3

Limits of Integrating OWL and Rules

As shown in [8], the availability of unrestricted role chains, inverse roles, concept conjunctions, and role conjunctions in the same DL fragment leads to a larger coverage of rules in the DL paradigm; by *unrestricted role chains* we mean that the set of complex role inclusions does not have to adhere to the regularity restrictions defined in [13]. Unfortunately, we show that often even minimal combinations of the mentioned constructors lead to undecidability, a result which significantly limits the scope of possible integrations of DLs and rules. In particular, we provide proofs of undecidability for the minimal DL fragments $\mathcal{ERI}\neg$, $\mathcal{ERI}(\sqcap,\sqcup)$, $\mathcal{ER}(\sqcap_R)$, $\mathcal{ERIA}(\sqcap)$, $\mathcal{ERIO}(\sqcap)$ and $\mathcal{ERI}_{\text{Self}}$.[1] For convenience, we will henceforth refer to these DLs as *the $\mathcal{ERI}$ fragments*.

As will be shown, we have that concept satisfiability from the first fragment and instance checking for the second can be reduced to the undecidable domino problem [5]. For the other remaining four we present a reduction from instance checking in the fragment to the problem of finding the intersection of two context-free grammars, also a well-known undecidable problem. All the proofs which state the undecidability of instance checking for a given fragment can be reduced to knowledge base satisfiability of the same fragment if the bottom concept $\bot$ is added.

From these results, we have that almost every extension of the fragment $\mathcal{ERI}(\sqcap)$ extended with any of the classical DL constructors leads to undecidability. To compensate for these negative results we propose at the end of the paper some restrictions other than regularity, which could perhaps be used to retain decidability.

The paper is structured as follows. After providing preliminary definitions and the formal semantics of the $\mathcal{ERI}$ fragments in Section 3.1, we give undecidability proofs for $\mathcal{ERI}\neg$ and $\mathcal{ERI}(\sqcap,\sqcup)$

[1] The symbol $\sqcap_R$ denotes role conjunction.

in Section 3.2. Section 3.3 contains undecidability proofs for the fragments $\mathcal{ER}(\sqcap_R)$, $\mathcal{ERIA}(\sqcap)$, $\mathcal{ERIO}(\sqcap)$ and $\mathcal{ERI}_{\text{Self}}$. In Section 3.4 we discuss possible options to retain decidability, and conclude the paper.

3.1 The $\mathcal{ERI}$ Fragments

Definition Let $\mathbf{C}$ be a set of *concept names* including a subset $\mathbf{N}$ of *nominals*, $\mathbf{R}$ a set of *role names*, and $\mathbf{I} = \{a, b, c, \dots\}$ a set of *individual names*. The set of *roles* is $\mathbf{R} \cup \{R^- | R \in \mathbf{R}\}$, where a role R^- is called the *inverse role* of R.

As usual, an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and an *interpretation function* $.^\mathcal{I}$ which associates, with each role name R, a binary relation $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, with the universal role U the universal relation $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, with each concept name C a subset $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$, where $C^\mathcal{I}$ is a singleton subset if $C \in \mathbf{N}$, and with each individual name a an element $a^\mathcal{I} \in \Delta^\mathcal{I}$.

Like in other DL fragments, we can divide the axioms in a knowledge base into an *ABox* $\mathcal{A}$, a *TBox* $\mathcal{B}$, and a *RBox* $\mathcal{R}$ statement[2] for any $\mathcal{ERI}$ fragment. A *TBox* is a finite set of *general concept inclusions* (GCIs) as depicted in Table 3.1. An *RBox* is a finite set of *role inclusion axioms* (RIAs), also depicted in Table 3.1. An *ABox* is a finite set of axioms of the form $C(a)$ or $R(a, b)$, where $C \in \mathbf{C}$ and $R \in \mathbf{R}$.

The formal model-theoretic semantics of the $\mathcal{ERI}$ fragments discussed in this paper is defined as expected along the lines from Table 3.1. The available constructors in a fragment can be easily derived using the letters and constructors that appear in its name. Note that all $\mathcal{ERI}$ fragments allow cyclic general concept inclusion and complex role inclusion axioms without imposing regularity restrictions.

Definition An interpretation $\mathcal{I}$ is a *model* of a given knowledge base KB if the conditions given in Table 3.1 are satisfied for all axioms in KB.

3.2 Reductions to the domino problem

In this section we provide undecidability proofs for the description logics $\mathcal{ERI}\neg$ and $\mathcal{ERI}(\sqcap, \sqcup)$. As in previous undecidability proofs for DLs [16] we reduce the (undecidable) domino problem to $\mathcal{ERI}\neg/\mathcal{ERI}(\sqcap, \sqcup)$ concept satisfiability.

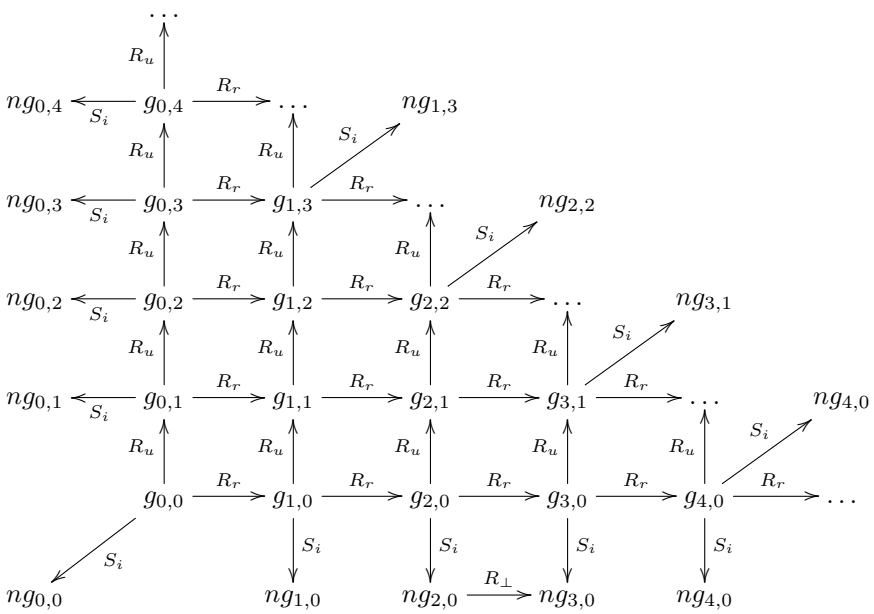
[2] Assertional, terminological, and role boxes respectively.

Table 3.1: Semantics of the $\mathcal{ERI}$ based DL Fragments

Constructor	**Name**	**Syntax**	**Semantics**
	Tbox Axiom (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
	Rbox Axiom (RIA)	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{E}$	Existential Restriction	$\exists R.C$	$\{\delta\vert$ there is ϵ with $\langle\delta,\epsilon\rangle \in R^{\mathcal{I}}$ and $\epsilon \in C^{\mathcal{I}}\}$
$\mathcal{R}$	Role Chain (RIA)	$R_1 \circ \ldots \circ R_n \sqsubseteq S$	$R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{I}$	Role Inverse	R^-	$\{\langle\delta,\epsilon\rangle\vert\langle\epsilon,\delta\rangle \in V^{\mathcal{I}}\}$
$\neg$	Concept Complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\sqcap$	Concept Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$\sqcup$	Concept Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\sqcap_R$	Role Conjunction	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
$\mathcal{A}$	Universal Restriction	$\forall R.C$	$\{\delta\vert$ for all ϵ with $\langle\delta,\epsilon\rangle \in R^{\mathcal{I}}$ we have $\epsilon \in C^{\mathcal{I}}\}$
$\mathcal{O}$	Nominal	{t}	$\{t\}^{\mathcal{I}}$
Self	Self Restriction	$\exists R.Self$	$\{\delta\vert\langle\delta,\delta\rangle \in R^{\mathcal{I}}\}$

where $C, D \in \mathbf{C}$, $R, S, U \in \mathbf{R}$, $t \in \mathbf{N}$

Figure 3.1: Graph for the proof of Theorem 3.2.1.



Definition A domino system $\mathcal{D} = (D, H, V)$ consists of a finite non-empty set of domino types $D = \{D_0, D_1, \ldots, D_n\}$, and sets of horizontally and vertically matching pairs $H \subseteq D \times D$ and $V \subseteq D \times D$. The problem is to determine if, for a given $\mathcal{D}$, there exists a tiling of an $\mathbb{N} \times \mathbb{N}$ grid such that each point of the grid is covered with a domino type in D and all horizontally and vertically adjacent pairs of domino types are in H and V respectively, i.e. a mapping

$$t : \mathbb{N} \times \mathbb{N} \to D \text{ such that for all } m, n \in \mathbb{N}, \langle t(m,n), t(m+1,n)\rangle \in H \text{ and}$$
$$\langle t(m,n), t(m,n+1)\rangle \in V$$

Given a domino system D, the problem of determining if there exists a tiling for $\mathcal{D}$ is known to be undecidable [5].

3.2.1 $\mathcal{ERI}(\sqcap, \sqcup)$ is undecidable

Theorem 3.2.1. *Checking satisfiability of $\mathcal{ERI}(\sqcap, \sqcup)_\perp$ and instance checking for $\mathcal{ERI}(\sqcap, \sqcup)$, without role regularity restrictions, are undecidable problems.*

Proof. See the graph in Figure 3.1 for an intuitive understanding of the interpretations considered in this proof–see the remarks made after the proof.

For any given domino instance problem $\mathcal{D}$, as defined in Definition 3.2, we define an $\mathcal{ERI}(\sqcap, \sqcup)_\perp$ knowledge base $KB_\mathcal{D}$, such that $\mathcal{D}$ has a solution if and only if some concept G defined in $KB_\mathcal{D}$ is satisfiable with respect to $KB_\mathcal{D}$. The axioms in $KB_\mathcal{D}$ are depicted in Table 3.2.

By definition, we have that if a concept G is satisfiable with respect to $KB_\mathcal{D}$, then there exists an interpretation $\mathcal{I}$ s.t. $\mathcal{I}$ is a model of $KB_\mathcal{D}$ and $G^\mathcal{I} \neq \emptyset$. From such an interpretation $\mathcal{I}$ we construct a solution for the domino instance problem $\mathcal{D}$.

Select an individual $a \in G^\mathcal{I}$ and define a function $f : \mathbb{N} \times \mathbb{N} \to \Delta^\mathcal{I}$, where $\Delta^\mathcal{I}$ is the domain of the interpretation $\mathcal{I}$ s.t. $f(1,1) = a$ and the following hold:

- $f(x+1,1) = a_r$ for some individual $a_r \in G^\mathcal{I}$ with $\langle f(x,1), a_r\rangle \in R_r$.
- $f(x,y+1) = a_u$ for some individual $a_u \in G^\mathcal{I}$ with $\langle f(x,y), a_u\rangle \in R_u$.

Since $\mathcal{I}$ is a model, we are guaranteed by axioms (3.3) and (3.4) that every $a \in G^\mathcal{I}$ has at least one R_r neighbor a_r and one R_u neighbor a_u, s.t. $a_r \in G^\mathcal{I}$ and $a_u \in G^\mathcal{I}$. Note that model $\mathcal{I}$ is not necessarily infinite.

By axiom (3.1) we have that $a \in G^\mathcal{I}$ implies $a \in C_i^\mathcal{I}$ for some i where $1 \leq i \leq n$. Axiom (3.2) implies that all the C_i concepts are pairwise disjoint. Therefore, for every $a \in G^\mathcal{I}$ we have that $a \in C_i^\mathcal{I}$ for a unique i where $1 \leq i \leq n$.

We define a function $t : \mathbb{N} \times \mathbb{N} \to D$, where D is the set of different tiles D_i s.t. $t(x,y) = D_i$ when $f(x,y) \in C_i^\mathcal{I}$. We will show now that the function t is a solution for the domino instance problem $\mathcal{D}$.

Since the interpretation $\mathcal{I}$ is a model, we have that there is no $\langle a, b\rangle \in R_\perp^\mathcal{I}$ with $b \in NG^\mathcal{I}$, by axiom (3.10). Therefore, we can conclude that there is neither $\langle a, b\rangle \in (S_i^- \circ R_r \circ S_j)^\mathcal{I}$ (role chains from axiom (3.8)) nor $\langle a, b\rangle \in (S_i^- \circ R_u \circ S_j)^\mathcal{I}$ (from axiom (3.9)) with $b \in NG$. Consequently we have that tiles D_i and D_j are horizontally (vertically) compatible if $a \in C_i$, $b \in C_j$, and $\langle a, b\rangle \in R_r^\mathcal{I}$ ($\langle a, b\rangle \in R_u^\mathcal{I}$).

We have

- $\langle f(x,y), f(x,w)\rangle \in R_u$ if $w = y+1$ and
- $\langle f(x,y), f(z,y)\rangle \in R_r$ if $z = x+1$ (by axiom 3.5).

To conclude, we have that the function f is defined for all possible pair combinations x and y of natural numbers. Every individual $f(x,y)$ is R_r- (R_u-) connected to another individual $f(x+1,y)$ ($f(x,y+1)$). R_r- (R_u-) connected individuals $f(x,y)$ and $f(z,w)$ are tile compatible, i.e. the tiles $t(x,y)$ and $t(z,w)$ are horizontally (vertically) compatible. Consequently, the function t is a solution for the domino instance problem $\mathcal{D}$.

Conversely, we need to derive an interpretation $\mathcal{I}$ which is a model with $G^{\mathcal{I}} \neq \emptyset$, from a mapping t which is a solution for the domino problem instance $\mathcal{D}$. The interpretation $\mathcal{I}$ is defined in Table 3.3.

We have that every individual $g_{x,y} \in G^{\mathcal{I}}$ is also in some $g_{x,y} \in C_i^{\mathcal{I}}$ (axiom (3.1)) and that the concepts $C_i^{\mathcal{I}}$ are pairwise disjoint (axiom (3.2)). Also, every individual $g_{x,y} \in G^{\mathcal{I}}$ has some $\langle g_{x,y}, g_{x+1,y}\rangle \in R_r$ and $\langle g_{x,y}, g_{x,y+1}\rangle \in R_u$ s.t. $g_{x+1,y} \in G^{\mathcal{I}}$ and $g_{x,y+1} \in G^{\mathcal{I}}$ (axioms (3.3) and (3.4)). All individuals $g_{x,y} \in C_i$ are in $\langle g_{x,y}, ng\rangle \in S_i$, s.t. $ng \in NG$ (axiom (3.6)). Since the individuals $g_{x,y}$ are connected through roles R_r and R_u, we have that axiom (3.5) does not produce any additional R_r connection.

$G^{\mathcal{I}}$ and $NG^{\mathcal{I}}$ are disjoint (axiom (3.7)). Since all $C_i^{\mathcal{I}}$ for every $g_{x,y} \in G^{\mathcal{I}}$ are derived from compatible tiles D_i (using the solution mapping t), we are guaranteed that axioms (3.8), (3.9), and (3.10) do not produce an inconsistency. Therefore, the interpretation $\mathcal{I}$ is a model for $KB_{\mathcal{D}}$ where $G \neq \emptyset$.

The proof just given shows that checking satisfiability for $\mathcal{ERI}(\sqcap, \sqcup)_{\perp}$ is indeed unsatisfiable. To reduce the proof to instance checking, we just need to apply some small changes. Substitute every appearance of $\perp$ by $\exists R_{\perp}.NG$ and add the RIAs

- $R \circ R_{\perp}.NG \sqsubseteq R_{\perp}$
- $R_{\perp}.NG \circ R \sqsubseteq R_{\perp}$
- $R_{\perp} \circ R_{\perp} \sqsubseteq R_{\perp}$
- $R_{\perp}^{-} \sqsubseteq R_{\perp}$

for all roles $R \cup R^{-} \in KB$.

Now we have that there is no solution to the domino instance problem $\mathcal{D}$ iff $KB_{\mathcal{D}} \models (\exists R_{\perp}.NG)(a)$. The roles introduced after the previous paragraph propagate any pair $\langle x, y\rangle \in R_{\perp}$ to every node in the grid, and therefore, if there is no solution for the domino instance problem $\mathcal{D}$, i.e. two tiles are incompatible, or some of the constraints defined in axioms (3.2) and (3.7) are not fulfilled, we have that $\langle a, x\rangle \in R$ and $x \in NG$. □ □

To return to the graph in Figure 3.1, note that all $g_{x,y}$ nodes need to be connected to their respective $ng_{x,y}$ for every model. In this particular graph we omit $ng_{2,1}$, $ng_{1,1}$, $ng_{1,2}$ to simplify the presentation. Through the S_i, where $1 \leq i \leq n$, from the connection from $g_{x,y}$ to the correspondent $ng_{x,y}$ we can infer the tile placed in the grid position $\langle x, y\rangle$ (note that the S_i connection is dependent on C_i, which is unique for every node).

In the graph we depict a connection between two right to left incompatible tiles, namely tiles in position (0, 2) and (0, 3). The $R_\perp$ connection is automatically derived from the role inclusion axioms (3.8) and (3.9) that are defined by the set of left to right compatible relations $H \in \mathcal{D}$. Therefore, the interpretation represented in this graph is not a model, i.e. cannot be mapped to a solution to the instance problem $\mathcal{D}$.

3.2.2 $\mathcal{ERI}\neg$ is Undecidable

Theorem 3.2.2. *Checking satisfiability for $\mathcal{ERI}\neg$, without role regularity restrictions, is undecidable.*

Proof. For any given domino system problem $\mathcal{D}$ we can construct an $\mathcal{ERI}\neg$ knowledge base $KB_\mathcal{D}$ containing a concept G such that G is satisfiable with respect to $KB_\mathcal{D}$ iff there exists a tiling for $\mathcal{D}$. $KB_\mathcal{D}$ axioms are depicted in Table 3.4.

Both $\top$ and $\perp$ concepts can be modeled in the DL fragment $\mathcal{ERI}\neg$. For any given concept C s.t. $C \in KB$ add the axioms

$$C \sqsubseteq \top \text{ and } \neg C \sqsubseteq \top.$$

To include the $\perp$ concept we add

$$\perp \sqsubseteq \neg\top$$

To ease the proof and without loss of generality we will directly make use of these concepts in the knowledge base $KB_\mathcal{D}$.

By definition, we have that if the concept G is satisfiable with respect to $KB_\mathcal{D}$, then there exists an interpretation $\mathcal{I}$ s.t. $\mathcal{I}$ is a model of $KB_\mathcal{D}$ and $G^\mathcal{I} \neq \emptyset$. From the interpretation $\mathcal{I}$ we construct a solution for the domino problem $\mathcal{D}$.

We select an individual $a \in G^\mathcal{I}$ and define a function $f : \mathbb{N} \times \mathbb{N} \to \Delta^\mathcal{I}$, where $\Delta^\mathcal{I}$ is the domain of the interpretation $\mathcal{I}$ s.t. $f(1,1) = a$ and the following hold.

- $f(x+1, 1) = a_r$ for only one individual $a_r \in G^\mathcal{I}$ s.t. $\langle f(x,1), a_r\rangle \in R_r$.
- $f(x, y+1) = a_u$ for only one individual $a_u \in G^\mathcal{I}$ s.t. $\langle f(x,y), a_u\rangle \in R_u$.

Since $\mathcal{I}$ is a model, we are guaranteed by axioms (3.11) and (3.12) that every $a \in G^\mathcal{I}$ has at least one R_r and one R_u neighbor a_r and a_u, respectively, s.t. $a_r \in G^\mathcal{I}$ and $a_u \in G^\mathcal{I}$. Note that the model $\mathcal{I}$ is not necessarily infinite.

We have that for every individual a in $\mathcal{I}$ either $a \in C_i^\mathcal{I}$ or $a \in \neg C_i^\mathcal{I}$ for every i where $1 \leq i \leq n$. We define a function $t : \mathbb{N}\times\mathbb{N} \to D$, where D is the set of different tiles D_i. We have that $t(x, y) = D_i$ if $j = 2^{k_1} + 2^{k_2} + \ldots + 2^{k_n} + 0$ s.t. $f(x, y) \in C_{k_1} \cup C_{k_2} \cup \ldots \cup C_{k_n}$. Note that $j = 0$ if we have

that $f(x,y) \in \neg C_0 \cup \neg C_1 \cup \cdots \cup \neg C_n$. We will see that the function t is a solution for the domino instance problem $\mathcal{D}$.[3]

By axioms (3.14) and (3.15) we have that if $a \in C_i$ then $\langle a,b\rangle \in R_{C_i}$ s.t. $b \in NG$. Conversely, we have that $\langle a,b\rangle \in R_{\neg R_{C_i}}$ if $a \in \neg C_i$ s.t. $b \in NG$. Therefore, for every individual $f(x,y)$ s.t. $t(x,y) = D_i$, we have that $\langle f(x,y), f(x,y)\rangle \in S_i^{\mathcal{I}}$ due to axioms (3.17) - (3.22).

Since the interpretation $\mathcal{I}$ is a model, we have that there is no $\langle a,b\rangle \in R_{\bot}^{\mathcal{I}}$ due to axiom (3.25). Therefore, we can conclude that there is neither $\langle a,b\rangle \in (S_i^- \circ R_r \circ S_j)^{\mathcal{I}}$ (role chains from axiom (3.23)) nor $\langle a,b\rangle \in (S_i^- \circ R_u \circ S_j)^{\mathcal{I}}$ (from axiom (3.24)). Consequently, we have that tiles D_i and D_j are horizontally (vertically) compatible if $\langle a,b\rangle \in R_r^{\mathcal{I}}$ ($\langle a,b\rangle \in R_u^{\mathcal{I}}$).

We furthermore have that

- $\langle f(x,y), f(z,y)\rangle \in R_r$ if $z = x+1$ (by axiom 3.13), and
- $\langle f(x,y), f(x,w)\rangle \in R_u$ if $w = y+1$.

To conclude, we have that the function f is defined for all possible pair combinations x and y of natural numbers. Every individual $f(x,y)$ is R_r-connected (respectively, (R_u)-connected) to another individual $f(x+1,y)$ (respectively, $f(x,y+1)$). R_r (respectively R_u) connected individuals $f(x,y)$ and $f(z,w)$ are tile compatible, i.e. the tiles $t(x,y)$ and $t(z,w)$ are horizontally (respectively vertically) compatible. Consequently, the function t is a solution for the domino instance problem $\mathcal{D}$.

Conversely, if $\mathcal{D}$ has a solution, then G is satisfiable with respect to $KB_{\mathcal{D}}$. From a tiling for $\mathcal{D}$ we can construct a model $\mathcal{I}$ as depicted in Table 3.5.

We have that every individual $g_{x,y} \in G^{\mathcal{I}}$ is also in $\langle g_{x,y}, g_{x+1,y}\rangle \in R_r$ and $\langle g_{x,y}, g_{x,y+1}\rangle \in R_u$ s.t. $g_{x+1,y} \in G^{\mathcal{I}}$ and $g_{x,y+1} \in G^{\mathcal{I}}$ (axioms (3.11) and (3.12)). All individuals $g_{x,y} \in (\neg)C_i \cup G$ are in $\langle g_{x,y}, ng_{x,y}\rangle \in R_{(\neg)C_i}$, s.t. $ng \in NG$ (axiom (3.13)). Also, all individuals $ng_{x,y} \in C_i \cup NG$ are in $\langle ng_{x,y}, ng_{x,y}\rangle \in R_{C_i}$, and respectively all $ng_{x,y} \in \neg C_i \cup NG$ are in $\langle ng_{x,y}, ng_{x,y}\rangle \in R_{\neg C_i}$. Due to the way in which the individuals $g_{x,y}$ are connected through roles R_r and R_u, we have that axiom (3.13) does not produce any R_r connection.

$G^{\mathcal{I}}$ and $NG^{\mathcal{I}}$ are disjoint (axiom (3.16)). Also, for every element $ng_{x,y} \in NG$, we have that $ng_{x,y} \in C_i$ if $g_{x,y} \in C_i$ and respectively, $ng_{x,y} \in \neg C_i$ if $g_{x,y} \in \neg C_i$. Roles S_i only connect an individual a with itself, or an individual $g_{x,y}$ with the correspondent $ng_{x,y}$ and vice versa. Since all $C_i^{\mathcal{I}}$ for every $g_{x,y} \in G^{\mathcal{I}}$ are derived from compatible tiles D_i (using the solution mapping t) we are guaranteed that axioms (3.17) - (3.25) do not produce an inconsistency. Therefore, the interpretation $\mathcal{I}$ is a model for $KB_{\mathcal{D}}$ where $G \neq \emptyset$.

[3] As an intuition, we have that the concepts C_i form a binary encoding that allow us to map each individual to a unique class.

□ □

The graphical representation for this second proof is quite similar to the previous one. We have that every node $g_{x,y}$ is connected to itself by an S_i, where $1 \leq i \leq n$, that uniquely determines the tile $D_i \in D$ it has to be mapped to. Again the set of roles (3.23) and (3.24) prune potential models that do not map to a solution to the domino instance problem $\mathcal{D}$.

3.3 Reductions to the intersection of two context free grammars

In this section we provide undecidability proofs for the description logic fragments $\mathcal{ER}(\sqcap_R)$, $\mathcal{ERIA}(\sqcap)$, $\mathcal{ERIO}(\sqcap)$, and $\mathcal{ERI}_{\text{Self}}$. As the title of the section states, we reduce the satisfiability problem of these fragments to the intersection problem for two context free grammars.

The proofs in this section are inspired by one which can be found in [28] related to query answering in $\mathcal{EL}^{++}$. By a reduction to the Post Correspondence Problem it is shown that this DL fragment cannot be extended with role conjunction and role inverses while retaining decidability.

Definition A context-free grammar G is defined as a 4-tuple $G = (V, \Sigma, R, S)$ where

- V is a finite set of non-terminals,
- Σ is a finite set of terminals, disjoint from V,
- R is a finite relation from V to $(V \cup \Sigma)^*$, and
- S is the start variable.

The language produced by a context free grammar is called a context free language. We assume, without loss of generality, that the free context grammars used in these proofs are in Chomsky normal form. Context free languages in this normal form only contain production rules of the forms

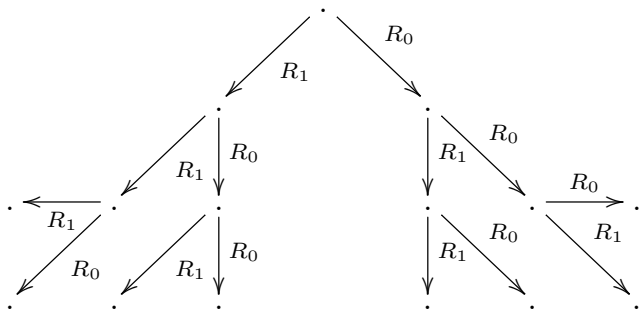
- $A \to BC$,
- $A \to \alpha$, and
- $S \to \epsilon$,[4]

where $A \cup B \cup C \in V$, $\alpha \in \Sigma$, and S is the start variable.

Given two free context languages L_1 and L_2 generated by two free context grammars L_{g1} and L_{g2}, the problem of determining if $L_1 \cap L_2 = \emptyset$ is known to be undecidable.

[4]Where ϵ stands for the empty string.

Figure 3.2: Graph for the proof of Theorem 3.3.1.



3.3.1 $\mathcal{ER}(\sqcap_R)$ is Undecidable

Theorem 3.3.1. *Inference checking for $\mathcal{ER}(\sqcap_R)$ and checking satisfiability for the description logic $\mathcal{ER}(\sqcap_R)_\bot$, without role regularity restrictions, are undecidable.*

Proof. In Figure 3.3.1 we provide a graph that depicts the intuition behind the structures of the models used in the proof; see also the remarks directly after the proof.

From a pair of context free grammars L_{g_1} and L_{g_2} we can can derive a $\mathcal{ER}(\sqcap_R)$ knowledge base KB and a concept E s.t. KB entails E iff the intersection of the languages L_1 and L_2, generated from the grammars L_{g_1} and L_{g_2}, is not empty.

Add a role R_{A_i} to KB for every $A \in V \in L_{g_i}$ and a role R_{α_i} for every $\alpha \in \Sigma \in L_{g_i}$ where $1 \leq i \leq 2$. Assuming Chomsky normal form, for every production rule of the form, $A \to \alpha \in L_{g_i}$ add an axiom $R_{\alpha_i} \sqsubseteq R_{A_i}$ to KB where $1 \leq i \leq 2$. For every $A \to BC \in L_{g_i}$ add an axiom $R_{B_i} \circ R_{C_i} \sqsubseteq R_{A_i}$ to KB where $1 \leq i \leq 2$.

Include GCIs $D \sqsubseteq \exists R.D$ for all roles $R_{\alpha_i} \in KB$ where $\alpha \in \Sigma \in L_{g_i}$ for $1 \leq i \leq 2$ and instantiate the concept D with an individual $a \in D$. If we have that $S \to \epsilon \in L_{g_i}$, we include $R_{S_i}(a, a)$ to the KB where $1 \leq i \leq 2$. Add axiom $R_{S_1} \sqcap R_{S_2} \sqsubseteq R_S$ where S_1 and S_2 are the start variables for the grammars L_{g1} and L_{g2}.

We have that $KB \models (\exists R_S.D)(a)$ iff $L_1 \cap L_2 \neq \emptyset$.

By the definition of D we have that all possible combinations of terminal symbols, represented by the roles R_{α_i}, start from the individual a, i.e. $KB \models \{a\} \sqsubseteq \exists R_{\alpha_{1i}}.\exists R_{\alpha_{2i}} \ldots \exists R_{\alpha_{ni}}.D$ for all sequences of $\alpha_{ji} \in \Sigma$. Due to the role inclusion axioms included in the KB, we have that $\langle x, y \rangle \in R_{S_i}$ iff we have that all simple roles $R_{\alpha i}$ in the path from x to y, where $\alpha \in \Sigma \in L_{g_i}$, form a word w s.t. $w \in L_1 \cup L_2$ if substituted by terminal symbols α where $1 \leq i \leq 2$.

By the combination of these two facts we have that there is an intersection of these two languages

if and only if $\exists R_S.D(a)$ is entailed by the knowledge base. Also, note that if both grammars contain the production rule $S \to \epsilon$ the latter statement is also the case, since $R_{S_1}(a,a) \cup R_{S_2}(a,a) \in KB$.

To reduce the problem to checking satisfiability for $\mathcal{ER}(\sqcap_R)_\bot$ we just add the axiom $\exists R_s.D \sqsubseteq \bot$. Note that the existence of any pair in $\langle x,y\rangle \in R_s$ even if $x \neq a$, already implies the existence of a non-empty intersection of languages L_1 and L_2. Since we have all possible combinations of non-terminals covered by the statement $D \sqsubseteq \exists R.D$, we have that $\langle x,y\rangle \in R_s$ is always a fact if the intersection of the languages L_1 and L_2 is non-empty.

Therefore, we have that KB is satisfiable iff $L_1 \cap L_2 = \emptyset$. □ □

The graph in Figure 3.3.1 corresponds to an interpretation derived from a context free language with the terminal symbols 1 and 0. As it can be easily seen, we have that all possible combinations of terminals, i.e. 0 and 1 in this case, are depicted in the intersection starting at any node. The role inclusion axioms added to the knowledge base capture all combination of terminals into non-terminals symbols and eventually find all words in the language, which will be represented through roles S_1 and S_2.

Note that, even though $\mathcal{ER}(\sqcap_R)_\bot$ is the smallest of the DL fragments considered in this paper, the undecidability result is not completely unintuitive. The three constructors available in $\mathcal{ER}(\sqcap_R)$ can be very easily reduced to the problem of the intersection for two context free grammars. As shown, the existential constructor is used to produce all possible combinations of terminal symbols, role chains can recognize all possible words from all these combinations, and role intersection can be used to check the emptiness of the intersection between languages.

3.3.2 $\mathcal{ERIA}(\sqcap)$ is Undecidable

Theorem 3.3.2. *Inference checking in $\mathcal{ERIA}(\sqcap)$ and checking satisfiability for the description logic $\mathcal{ERIA}(\sqcap)_\bot$, without role regularity restrictions, is undecidable.*

Proof. The undecidability of $\mathcal{ERIA}(\sqcap)$ can be easily derived from the results in Section 3.3.1. We create a knowledge base in the same way as in the previous proof but instead of including $R_{S_1} \sqcap R_{S_2} \sqsubseteq R_S$ and $D(a)$, we add statements $(\forall R_{S_1}.E)(a)$, $(\forall R_{S2}.F)(a)$, $E \sqcap F \sqsubseteq G$, and $G \sqsubseteq \forall R^-_{S_1}.G$.

We have that $KB \models G(a)$ iff there $L_1 \cap L_2 \neq \emptyset$.

Note that the existence of an individual $x \in G$ implies the existence of the pair $\langle a,x\rangle \in R_{S_1} \cap R_{S_2}$. As defined in Section 3.3.1, this entails $L_1 \cap L_2 \neq \emptyset$. By the last axiom $G \sqsubseteq \forall R^-_{S_1}.G$, we have that the existence of $x \in G$ implies $a \in G$. Therefore we have that $KB \models G(a)$ if and only if $L_1 \cap L_2 \neq \emptyset$.

To reduce concept satisfiability to the case just shown, we produce a knowledge base KB_s by removing $G \sqsubseteq \forall R^-_{S_1}.G$ from the previous one and add the *Tbox* statement $G \sqsubseteq \bot$.

We have that KB_s is satisfiable iff $L_1 \cap L_2 \neq \emptyset$. As stated above, the existence of an individual $x \in G$ implies the existence of two words S_1 and S_2 from a to x and therefore the non-emptyness of the intersection of languages L_1 and L_2. Since all combinations of words start at a we have that there will always be an $x \in G$ if there exists an intersection for the languages. □ □

3.3.3 $\mathcal{ERIO}(\sqcap)_\bot$ is Undecidable

Theorem 3.3.3. *Inference checking in $\mathcal{ERIO}(\sqcap)$ and checking satisfiability for $\mathcal{ERIO}(\sqcap)_\bot$, without role regularity restrictions, is undecidable.*

Proof. Again, the undecidability results for $\mathcal{ERIO}(\sqcap)$ can be easily deduced from the previous ones. We build a knowledge base KB in the same way as shown in Section 3.3.1, without including axiom $R_{S_1} \sqcap R_{S_2} \sqsubseteq R_S$. We have that $KB \models \exists(R_{S_1} \circ R_{S_2}^-).\{a\}(a)$ if and only if $L_1 \cap L_2 \neq \emptyset$. Note that *ABox* statements $\exists(R_{S_1} \sqcap R_{S_2}).\top(a)$[5] and $\exists(R_{S_1} \circ R_{S_2}^-).\{a\}(a)$ are equivalent and therefore the arguments in the $\mathcal{ER}(\sqcap_R)$ proof can be reused.

To reduce concept satisfiability, add the concept inclusion $\{a\} \sqcap \exists(R_{S_1} \circ R_{S_2^-}).\{a\} \sqsubseteq \bot$. We have that KB is satisfiable if and only if $L_1 \cap L_2 = \emptyset$. □ □

3.3.4 $\mathcal{ERI}_{\mathbf{Self}}$ is Undecidable

Theorem 3.3.4. *Inference checking in $\mathcal{ERI}_{Self}$ and checking satisfiability for $\mathcal{ERI}_{Self\bot}$, without role regularity restrictions, is undecidable.*

Proof. We can reuse the previous proof by substituting the *ABox* statement $\exists(R_{S_1} \circ R_{S_2}^-).\{a\}(a)$ by the equivalent $\exists(R_{S_1} \circ R_{S_2}^-).\text{Self}(a)$. □ □

3.3.5 Remarks to the proofs

Some of the proofs in this section are based on inferencing tasks with non-atomic concepts C. Is easy to see that all of these can be reduced to inference tasks only referring to atomic concepts C by adding auxiliary general concept inclusion axioms $C \sqsubseteq D$ or role inclusion axioms $R \sqsubseteq S$ with fresh concepts D and roles S.

Also, note that the inclusion of some kind of negation, such as the $\bot$ concept, is required to reduce the satisfiability problem. Otherwise we are not able to prune any potential models that are not derived from instances of the problem with (respectively, without) a solution.

[5] This is not the exact axiom used in the $\mathcal{ER}(\sqcap_R)$ proof, since $\top$ is not allowed, but it would still make the argument valid.

Note furthermore that, even if it is not stated explicitly, all proofs consider the case where both languages may produce the empty string, as it is defined for the $\mathcal{ER}(\sqcap_R)$ undecidability proofs.

3.4 Conclusions and Future Work

We have shown in this paper that extending $\mathcal{ERI}(\sqcap)$ with most of the classic DL constructors results in undecidability if the fragment does not enforce role regularity constraints. The results indicate severe limits for potential integrations between the most expressive DLs and rules approaches, since the latter do not need to take into account these kinds of constraints.

However, the negative results presented also raise the question if there are other restrictions than role regularity that can be imposed to retain decidability. As future work we intend to study some of the ideas presented in the next paragraphs.

We could attempt to restrict cycles over the existential quantifiers s.t. $A \sqsubseteq^* \exists R.A$ where $\sqsubseteq^*$ is the transitive closure of the $\sqsubseteq$ operator. This would limit the fragment to only produce finite models and retrieve decidability.

We could also restrict other constructors, such as constraining role conjunction only to intersect two roles if at most one of them is complex. The Self constructor can always be restricted to simple roles, i.e. $\exists R.\text{Self}$ for only simple roles S and nominals can be defined to only occur on the left hand side or the right hand side of concept inclusions.

Other routes have also been taken in the literature, which could be used to augment our setting, e.g. using nominal schemas [26] or by incorporating ideas from existential rules [27].

Table 3.2: $KB_{\mathcal{D}}$ axioms

$$G \sqsubseteq C_0 \sqcup \ldots \sqcup C_n \tag{3.1}$$

where n is the number of different tiles $D_i \in D$

$$C_i \sqcap C_j \sqsubseteq \bot \tag{3.2}$$

for all possible combinations of i and j s.t. $i \neq j$

$$G \sqsubseteq \exists R_r.G \tag{3.3}$$

$$G \sqsubseteq \exists R_u.G \tag{3.4}$$

$$R_u^- \circ R_r \circ R_u \sqsubseteq R_r \tag{3.5}$$

$$C_i \sqsubseteq \exists S_i.NG \tag{3.6}$$

$$NG \sqcap C_i \sqsubseteq \bot \tag{3.7}$$

for $1 \leq i \leq n$

$$S_i^- \circ R_r \circ S_j \sqsubseteq R_\bot \tag{3.8}$$

for all S_i and all S_j s.t. that $\langle D_i, D_j \rangle \notin H$

$$S_i^- \circ R_u \circ S_j \sqsubseteq R_\bot \tag{3.9}$$

for all S_i and all S_j s.t. that $\langle D_i, D_j \rangle \notin V$

$$\exists R_\bot.NG \sqsubseteq \bot \tag{3.10}$$

Table 3.3: Interpretation $\mathcal{I}$

$\Delta^{\mathcal{I}} =$	$\{g_{x,y} \mid x, y \in \mathbb{N}\} \cup \{b\}$	$R_r^{\mathcal{I}} =$	$\{\langle g_{x,y}, g_{z,y} \rangle \mid x = z + 1\}$
$G^{\mathcal{I}} =$	$\{g_{x,y} \mid x, y \in \mathbb{N}\}$	$R_u^{\mathcal{I}} =$	$\{\langle g_{x,y}, g_{x,w} \rangle \mid w = y + 1\}$
$NG^{\mathcal{I}} =$	$\{ng\}$	$S_i^{\mathcal{I}} =$	$\{\langle g_{x,y}, ng \rangle \in S_i \mid g \in C_i\}$
$C_i^{\mathcal{I}} =$	$\{g_{x,y} \in \Delta^{\mathcal{I}} \mid t(x,y) = D_i\}$		

Table 3.4: $KB_{\mathcal{D}}$ axioms

$$G \sqsubseteq \exists R_r.G \tag{3.11}$$

$$G \sqsubseteq \exists R_u.G \tag{3.12}$$

$$R_u^- \circ R_r \circ R_u \sqsubseteq R_r \tag{3.13}$$

$$C_i \sqsubseteq \exists R_{C_i}.NG \tag{3.14}$$

$$\neg C_i \sqsubseteq \exists R_{\neg C_i}.NG \tag{3.15}$$

where $0 \leq i \leq n$ where $n = \log_2 k - 1$ s.t. k is the number of different tiles $D_i \in D$.*

$$NG \sqsubseteq \neg G \tag{3.16}$$

$$R_{\neg C_n} \circ R_{\neg C_n}^- \circ \cdots \circ R_{\neg C_1} \circ R_{\neg C_1}^- \circ R_{\neg C_0} \circ R_{\neg C_0}^- \sqsubseteq S_0 \tag{3.17}$$

$$R_{\neg C_n} \circ R_{\neg C_n}^- \circ \cdots \circ R_{\neg C_1} \circ R_{\neg C_1}^- \circ R_{C_0} \circ R_{C_0}^- \sqsubseteq S_1 \tag{3.18}$$

$$R_{\neg C_n} \circ R_{\neg C_n}^- \circ \cdots \circ R_{C_1} \circ R_{C_1}^- \circ R_{\neg C_0} \circ R_{\neg C_0}^- \sqsubseteq S_2 \tag{3.19}$$

$$R_{\neg C_n} \circ R_{\neg C_n}^- \circ \cdots \circ R_{C_1} \circ R_{C_1}^- \circ R_{C_0} \circ R_{C_0}^- \sqsubseteq S_3 \tag{3.20}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \tag{3.21}$$

$$R_{C_n} \circ R_{C_n}^- \circ \cdots \circ R_{C_1} \circ R_{C_1}^- \circ R_{C_0} \circ R_{C_0}^- \sqsubseteq S_n \tag{3.22}$$

$$S_i \circ R_r \circ S_j \sqsubseteq R_\perp \tag{3.23}$$

for all S_i and all S_j s.t. that $\langle D_i, D_j \rangle \notin H$

$$S_i \circ R_u \circ S_j \sqsubseteq R_\perp \tag{3.24}$$

for all S_i and all S_j s.t. that $\langle D_i, D_j \rangle \notin H$

$$\exists R_\perp.\top \sqsubseteq \perp \tag{3.25}$$

* Note that, if $\log_2 k$ is not a natural number we can just add repetitions of the existing tiles in $\mathcal{D}$ until we have that the number of tiles is a power of 2.

How to create axioms (3.17 - 3.22): if the ith digit of the binary representation of number j is a 0, where S_j is on the right hand side of the axiom, then chain the pair of roles $R_{\neg C_i} \circ R_{\neg C_i}^-$ to the left hand side of the axiom. If the ith digit is a 1, then chain $R_{C_i} \circ R_{C_i}^-$ (any order works as long as pairs are together).

Table 3.5: Interpretation $\mathcal{I}$

$$
\begin{aligned}
\Delta^{\mathcal{I}} =& \{g_{x,y} | x, y \in \mathbb{N}\} \cup \{ng_{x,y} | x, y \in \mathbb{N}\} \\
G^{\mathcal{I}} =& \{g_{x,y} | x, y \in \mathbb{N}\} \\
NG^{\mathcal{I}} =& \{ng_{x,y} | x, y \in \mathbb{N}\} \\
C_i^{\mathcal{I}} =& \{g_{x,y} | t(x,y) = D_j \text{ s.t. } k = 1 \text{ where } k \text{ is the } ith \text{ binary digit of } j\} \\
& \cup \{ng(x,y) | g(x,y) \in C_i\} \\
R_r^{\mathcal{I}} =& \{\langle g_{x,y}, g_{z,y}\rangle | x = z + 1\} \\
R_u^{\mathcal{I}} =& \{\langle g_{x,y}, g_{x,w}\rangle | w = y + 1\} \\
R_{C_i}^{\mathcal{I}} =& \{\langle g_{x,y}, ng_{x,y}\rangle | g_{x,y} \in C_i\} \cup \{\langle ng_{x,y}, ng_{x,y}\rangle | ng_{x,y} \in C_i\} \\
R_{\neg C_i}^{\mathcal{I}} =& \{\langle g_{x,y}, ng_{x,y}\rangle | g_{x,y} \in \neg C_i\} \cup \{\langle ng_{x,y}, ng_{x,y}\rangle | ng_{x,y} \in \neg C_i\} \\
S_i^{\mathcal{I}} =& \{\langle ng_{x,y}, ng_{x,y}\rangle, \langle ng_{x,y}, g_{x,y}\rangle, \langle g_{x,y}, ng_{x,y}\rangle, \langle g_{x,y}, g_{x,y}\rangle | t(x,y) = D_i\}
\end{aligned}
$$

4

A Syntax proposal for Nominal Schemas

Nominal schemas [21; 26] are a new description-logic style extension of OWL 2 [34] which can be used like "variable nominal classes" within OWL 2 axioms. Although their semantics restricts them only to stand for named individuals, nominal schemas allows us to express arbitrarily shaped (Datalog) rules within the description logic (DL) paradigm, hence pushing the expressivity of OWL 2 DL and its fragments even further.

While the semantic intuition behind nominal schemas is the same as the one behind DL-safe variables presented in [32], the difference lies in the fact that DL-safe variables are tied to rule languages, while nominal schemas integrate seamlessly with DL syntax. The proposed extension encompasses DL-safe variable SWRL [15; 33; 21] while staying within the DL/OWL language paradigm and without employing hybrid approaches.

Nominal schemas have been introduced as a new general constructor for DL, denoted by the letter $\mathcal{V}$ in the DL nomenclature. The addition of nominal schemas has been considered for several DLs such as $\mathcal{SROIQ}$ that underlies OWL 2 DL, and $\mathcal{SROEL}$ that underlies the OWL 2 EL profile (define DL $\mathcal{SROIQV}$ and $\mathcal{SROELV}$, respectively, as extensions of the DLs $\mathcal{SROIQ}$ and $\mathcal{SROEL}$). It has been shown in [26] that the worst-case complexity of $\mathcal{SROIQV}$ remains N2EXPTIME-complete, i.e., no worse than $\mathcal{SROIQ}$. Furthermore, in the same paper, a tractable fragment of $\mathcal{SROIQV}$ has been identified. This fragment is called $\mathcal{SROELV}_n$ which is obtained by extending $\mathcal{SROEL}$ with nominal schema in a slightly restricted form. Nevertheless, it still covers[1] both OWL 2 EL and (DL-safe) OWL 2 RL.

We present an example of nominal schemas in the following. First, rules such as (4.1) are not

[1] without datatype-related features

expressible in the current OWL 2 DL standard.

$$\text{hasFather}(x,y) \wedge \text{hasBrother}(y,z) \wedge \text{hasTeacher}(x,z) \rightarrow \text{ChildTaughtByUncle}(x) \tag{4.1}$$

Intuitively, this is due to the fact that the body of the above rules is not tree-shaped (see [21] for further discussion about this). In contrast, using nominal schemas, rule (4.1) can be expressed as (4.2).

$$\exists \text{hasTeacher}.\{z\} \sqcap \exists \text{hasFather}.\exists \text{hasBrother}.\{z\} \sqsubseteq \text{ChildTaughtByUncle}. \tag{4.2}$$

The expression $\{z\}$ is a nominal schema, which is to be read as a variable nominal that can only represent nominals (i.e., z binds to known individuals), where the binding is the same for all occurrences of the nominal schema in an axiom. Variables x and y can still take arbitrary values and are hidden in the DL syntax, z needs to be restricted to be DL-safe to retain the conclusion. For a more detailed description of nominal schemas including their formal semantics see [26].

This document proposes different ways to represent nominal schemas within prominent variants of OWL syntax: Functional, Manchester, Turtle and RDF/XML. For an introduction of the OWL syntax, consult [34]. Mapping from Turtle triples to RDF/XML is a well defined and automatized process so the RDF/XML based syntax will not be directly addressed in this document, it is assumed that it can be easily derived from the Turtle Syntax.

New reserved words are presented to mark the appearance of nominal schemas in the different syntax variants (Functional, Manchester and Turtle) as well as the necessary modifications to their grammars (Functional and Manchester). The representation of nominal schemas in Turtle syntax is defined by the mappings from Functional and Manchester.

Several approaches were considered for the representation and storage of nominal schemas, such as the use of entities with the ontology namespace, but this paper proposes the use of string literals. With this approach, we prevent the possible overlap that could be produced by giving the same name to two different nominal schemas. If these are declared as entities and, by error, two of them share the same name, they will end up pointing to the same node in an RDF graph when they most likely refer to different individuals.

The selected approach, the use of a xsd:string datatype, is also considered by the RIF XML format [35]. Note that the same nominal schema can never appear in two different statements of an ontology—more precisely, if the same variable occurs in different axioms, then they are considered distinct (i.e., local to the axiom), in a way similar to the use of variables in rules. So nominal schema is local to one single axiom. By using a string type, the occurrence of the nominal schema is exclusively bound to the axiom where it appears and the same string could be repeated in different axioms along the ontology safely. Even if two nominal schemas use the same string, they will be

considered as different occurrences of a datatype and therefore, they can be understood as two separated nodes in an RDF graph.

Using the underscore to mark the appearance of a nominal schema, as it is done for Turtle blank nodes, was also considered. This approach was rejected because it could induce errors. Although in some cases both nominal schemas and blank nodes can represent individuals in an RDF graph they are completely different concepts. Using the underscore to mark both could be tricky and would make mappings from and to Turtle syntax difficult to define. With such a similar syntax the mapping may produce errors confusing nominal schemas with blank nodes and problems may arise when we want to move from the Turtle syntax to an RDF Graph.

The document is structured as follows. Section 4.1 contains the necessary modifications that have to be made to the Manchester and Functional Syntax grammars in order to include nominal schemas. Section 4.2 refers to the mappings from these syntaxes to Turtle. Section 4.3 concludes. Appendix 4.4 contains two examples for the usage of nominal schema in the different syntax variants that are discussed in the document.

4.1 Grammar Modifications

We propose several changes to the grammars of the different OWL syntaxes in order to include nominal schemas. The presented changes are designed to be minimal and imply very small modifications to the formal definitions of these grammars.

Functional Syntax Grammar Modifications

We define in this section the required modifications we propose for the Functional Syntax grammar [6]. The reserved word ObjectVariable will be used to mark the appearance of a nominal schema. The nominal schema will be in parentheses and will always be followed by the expression '^^xsd:string'. The proposed changes are as follows.

Add the non-terminal symbol **ObjectVariable**, to the **ClassExpression** production rule:

ClassExpression := **Class** | **ObjectIntersectionOf** | **ObjectUnionOf** |
ObjectComplementOf | **ObjectOneOf** |
ObjectSomeValuessFrom | **ObjectAllValuesFrom** |
ObjectHasValue | **ObjectHasSelf** |
ObjectMinCardinality | **ObjectMaxCardinality** |
ObjectExactCardinality | **DataSomeValuesFrom** |
DataAllValuesFrom | **DataHasValue** |
DataMinCardinality | **DataMaxCardinality** |
DataExactCardinality | **ObjectVariable**

Add the next production rule to the grammar:

ObjectVariable := 'ObjectVariable (' **quotedString**'^^xsd:string)'

Although nominal schemas are not conceptually class expressions, their addition in this part of the grammar has been chosen in order to keep the modifications as small as possible.

Manchester Syntax Grammar Modifications

Again, the reserved word ObjectVariable will be used to mark the appearance of the nominal schemas in the Manchester Syntax [12]. As in the Functional Syntax, the nominal will be in parentheses and followed by '^^xsd:string'. The needed changes to this grammar are:

Add the non-terminal symbol **ObjectVariable** to the **atomic** production rule:

atomic := **classIRI** | '{'**individualList**'}' | '('**description**')' |
ObjectVariable

Add the next production rule to the grammar:

ObjectVariable := 'ObjectVariable (' **quotedString**'^^xsd:string)'

4.2 Mapping FS and MS to Turtle

We define the syntax of nominal schemas in Turtle through the mapping from Functional and Manchester Syntaxes to the triple-notation. We assume that from this notation the process to move

to RDF/XML is already formalized so, as said before, the XML syntax will not be directly addressed in this document.

Functional Syntax to and from Turtle

The W3C document containing the formal mapping from FS to Turtle can be found in [31]. To add nominal schemas syntax to the mappings, first add the next row to the mapping from FS to Turtle:

Functional-Style Syntax	S Triples Generated in an Invocation of T(S)	Main Node of T(S)
ObjectVariable(”v1”^^xsd:string)	_:x rdf:type owl:ObjectVariable _:x owl:variableId ”v1”	_:x

Then add the next row to the mapping from Turtle to FS:

RDF/XML Triples	Functional Syntax
_:x rdf:type owl:ObjectVariable _:x owl:variableId ”v1”	ObjectVariable(”v1”^^xsd:string)

Manchester Syntax to and from Turtle

The mappings between Manchester Syntax and Turtle are defined in a similar way as the one from the Functional Syntax. To include the nominal schema in this mapping, we first need to add the next row to the table from MS to Turtle:

Manchester-Style Syntax	S Triples Generated in an Invocation of T(S)	Main Node of T(S)
Variable ”v1”^^xsd:string	_:x rdf:type owl:ObjectVariable _:x owl:variableId ”v1”	_:x

Then add the next row to the mapping from Turtle to FS:

RDF/XML Triples	Manchester Syntax
_:x rdf:type owl:ObjectVariable _:x owl:variableId ”v1”	Variable ”v1”^^xsd:string

4.3 Conclusions

In this document we propose ways for representing nominal schemas in the different syntaxes of the OWL language. Reserved words have been provided for Functional, Manchester, Turtle and RDF/XML syntaxes, along with the consistent modifications to their grammars and mapping functions. Nominal schemas will be stored as string values in the OWL syntaxes to prevent overlapping errors. In the appendix of this document two examples are presented showing nominal schemas across the different covered syntaxes of OWL.

Acknowledgements: This work was partially supported by the National Science Foundation under award 1017225 "III: Small: TROn—Tractable Reasoning with Ontologies." The first author acknowledges support from Programa de Intercambio de la Universidad Pontificia de Salamanca 2010/11. The second author acknowledges support by a Fulbright Indonesia Presidential Scholarship PhD Grant 2010.

4.4 Syntax Examples

4.4.1 Example 1

Rule Syntax

$\text{hasFather}(x, y) \wedge \text{hasBrother}(y, z) \wedge \text{hasTeacher}(x, z) \wedge \rightarrow \text{ChildTaughtByUncle}(x)$

DL Syntax

$\exists \text{hasFather}.(\exists \text{hasBrother}.\{z\}) \sqcap \exists \text{hasTeacher}.\{z\} \sqsubseteq \text{ChildTaughtByUncle}$

Functional Syntax

```
SubClassOf(

ObjectIntersectionOf(
    ObjectSomeValuesFrom( :hasFather
        ObjectSomeValuesFrom( :has Brother ObjectVariable("v1"^^xsd:string) ))
    ObjectSomeValuesFrom( :hasTeacher ObjectVariable("v1"^^xsd:string) )
)
```

:ChildTaughtByUncle
)

RDF/XML Syntax

_:x1 rdfs:subClassOf :ChildTaughtByUncle

_:x1 rdf:type owl:Class
_:x1 owl:intersectionOf (_:x2 _:x3)

_:x2 rdf:type owl:Restriction
_:x2 owl:onProperty :hasFather
_:x2 owl:someValuesFrom _:x5

_:x3 rdf:type owl:Restriction
_:x3 owl:onProperty :hasTeacher
_:x3 owl:someValuesFrom _:x4

_:x4 rdf:type owl:Restriction
_:x4 owl:onProperty :hasBrother
_:x4 owl:someValuesFrom :x6

_:x6 rdf:type owl:ObjectVariable
_:x6 owl:variableId "v1"

_:x5 rdf:type owl:ObjectVariable
_:x5 owl:variableId "v1"

Manchester Syntax

Class: ChildTaughtByUncle
SubClassOf:
(hasTeacher some (Variable "v1"^^xsd:string))
and
(hasSubmittedPaper some
(hasFather some (hasBrother some (Variable "v1"^^xsd:string))))

4.4.2 Example 2

Rule Syntax

hasReviewAssignment(v, x) $\wedge$ hasAuthor(x, y) $\wedge$ atVenue(x,z) $\wedge$
hasSubmittedPaper(v, u) $\wedge$ hasAuthor(u, y) $\wedge$ atVenue(u, z)
$\rightarrow$ ReviewerWithConflictingAssignment(v)

DL Syntax

$\exists$hasReviewAssignment.($\exists$hasAuthor.$\{a\} \sqcap \exists$atVenue.$\{b\}) \sqcap$
$\exists$hasSubmittedPaper.($\exists$hasAuthor.$\{a\} \sqcap \exists$atVenue.$\{b\}$)
$\sqsubseteq$ ReviewerWithConflictingAssignment

Functional Syntax

```
SubClassOf(

ObjectIntersectionOf(
    ObjectSomeValuesFrom ( :hasReviewAssign ObjectIntersectionOf (
            ObjectSomeValuesFrom (:hasAuthor ObjectVariable("v1"^^xsd:string))
            ObjectSomeValuesFrom (:atVenue    ObjectVariable("v2"^^xsd:string)) )
    )
    ObjectSomeValuesFrom ( :hasSubmittedPaper ObjectIntersectionOf (
            ObjectSomeValuesFrom (:hasAuthor ObjectVariable("v1"^^xsd:string))
            ObjectSomeValuesFrom (:atVenue    ObjectVariable("v2"^^xsd:string)) )
    )
)

:ReviewerWithConflictingAssignment
)
```

RDF/XML Syntax

```
_:x1 rdfs:subClassOf :ReviewerWithConflictingAssignment

_:x1 rdf:type owl:Class
```

_:x1 owl:intersectionOf (_:x2 _:x3)

_:x2 rdf:type owl:Restriction	_:x3 rdf:type owl:Restriction
_:x2 owl:onProperty :hasReviewAssign	_:x3 owl:onProperty :hasSubmittedPaper
:x2 owl:intersectionOf (:x4 _:x5)	_:x3 owl:intersectionOf (_:x8 _:x9)
_:x4 rdf:type owl:Restriction	_:x8 rdf:type owl:Restriction
_:x4 owl:onProperty :hasAuthor	_:x8 owl:onProperty :hasAuthor
_:x4 owl:someValuesFrom _:x6	_:x8 owl:someValuesFrom :x10
_:x6 rdf:type owl:ObjectVariable	_:x10 rdf:type owl:ObjectVariable
_:x6 owl:variableId "v1"	_:x10 owl:variableId "v1"
_:x5 rdf:type owl:Restriction	_:x9 rdf:type owl:Restriction
_:x5 owl:onProperty :atVenue	_:x9 owl:onProperty :atVenue
_:x5 owl:someValuesFrom _:x7	_:x9 owl:someValuesFrom :x11
_:x7 rdf:type owl:ObjectVariable	_:x11 rdf:type owl:ObjectVariable
_:x7 owl:variableId "v2"	_:x11 owl:variableId "v2"

Manchester Syntax

Class: ReviewerWithConflictingAssignment

SubtClassOf:

(hasReviewAssign some

((hasAuthor some (Variable "v1"^^xsd:string)) and (atVenue some (Variable "v2"^^xsd:string))))

and

(:hasSubmittedPaper some

((hasAuthor some (Variable "v1"^^xsd:string)) and (atVenue some (Variable "v2"^^xsd:string))))

References

Jürgen Angele. OntoBroker – mature and approved semantic middleware. *Semantic Web journal*, 2013. to appear. Available from http://www.semantic-web-journal.net/.

Krzysztof R. Apt. *From Logic Programming to Prolog.* International Series in Computer Science. Prentice Hall, 1997.

F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, second edition, 2007.

Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2nd edition, 2007.

R. Berger. The undecidability of the dominoe problem. In *Mem. Amer. Math. Soc. 66*, 1966.

Peter F. Patel-Scheneider Boris Motik and Bijan Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style.* W3C Recommendation 27 October 2009, 2009. http://www.w3.org/TR/owl2-syntax/.

D. Carral Martínez, A. Krisnadhi, F. Maier, K. Sengupta, and P. Hitzler. Reconciling OWL and rules. Technical report, Kno.e.sis Center, Wright State University, Dayton, Ohio, U.S.A., 2011. Available from `http://www.pascal-hitzler.de/`.

David Carral Martínez and Pascal Hitzler. Extending description logic rules. In Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications – 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, Lecture Notes in Computer Science, pages 345–359. Springer, 2012.

P. Hitzler and B. Parsia. Ontologies and rules. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 111–132. Springer, 2 edition, 2009.

Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation 27 October 2009, 2009. Available from http://www.w3.org/TR/owl2-primer/.

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

Matthew Horridge and Peter F. Patel-Scheneider. *OWL 2 Web Ontology Language: Manchester Syntax*. W3C Working Group Note 27 October 2009, 2009. http://www.w3.org/TR/owl2-manchester-syntax/.

I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.

I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL Rules: A proposal and prototype implementation. *J. of Web Semantics*, 3(1):23–40, 2005.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission 21 May 2004, 2004. Available from http://www.w3.org/Submission/SWRL/.

Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Artificial Intelligence*, pages 79–104, 2004.

Michael Kifer and Harold Boley, editors. *RIF Overview*. W3C Working Group Note 22 June 2010, 2010. Available from http://www.w3.org/TR/rif-overview/.

M. Knorr, P. Hitzler, and F. Maier. Reconciling OWL and non-monotonic rules for the Semantic Web. Technical report, Kno.e.sis Center, Wright State University, Dayton, OH, U.S.A., 2011. Available from `http://www.pascal-hitzler.de/`.

Matthias Knorr, Pasal Hitzler, and Frederick Maier. Reconciling OWL and non-monotonic rules for the Semantic Web. In *Proceedings ECAI2012*, 2012. to appear.

A. Krisnadhi and P. Hitzler. A tableau algorithm for description logics with nominal schemas. Technical report, Kno.e.sis Center, Wright State University, Dayton, OH, U.S.A., 2011. Available from `http://www.pascal-hitzler.de/`.

Adila Krisnadhi, Frederick Maier, and Pascal Hitzler. OWL and rules. In A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, and P.F. Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data. 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 382–415. Springer, Heidelberg, 2011.

Adila Krisnadhi, Frederick Maier, and Pascal Hitzler. OWL and Rules. In Axel Polleres et al., editors, *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 382–415. Springer, Heidelberg, 2011.

M. Krötzsch, S. Rudolph, and P. Hitzler. Description logic rules. In M. Ghallab et al., editors, *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008*, pages 80–84. IOS Press, 2008.

M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In A. Sheth et al., editors, *Proc. of the 7th International Semantic Web Conference (ISWC-08)*, volume 5318 of *Lecture Notes in Computer Science*, pages 649–664. Springer, 2008.

Markus Krötzsch. *Description Logic Rules*, volume 008 of *Studies on the Semantic Web*. IOS Press/AKA, 2010.

Markus Krötzsch, Frederick Maier, Adila A. Krisnadhi, and Pascal Hitzler. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In S. Sadagopan, Krithi Ramamritham, Arun Kumar, M.P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th International World Wide Web Conference, WWW2011, Hyderabad, India, March/April 2011*, pages 645–654. ACM, New York, 2011.

Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*, pages 963–968. AAAI Press/IJCAI, 2011.

Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive Queries for a Tractable Fragment of OWL 1.1. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudr-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference (ISWC 2007), Busan, Korea, November 11-15, 2007*, volume 4825 of *LNCS*, pages 310–323. Springer, 2007.

Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2012.

B. Motik, U. Sattler, and R. Studer. Query Answering for OWL DL with Rules. *J. of Web Semantics*, 3(1):41–60, 2005.

Boris Motik and Peter F. Patel-Scheneider, editors. *OWL 2 Web Ontology Language: Mapping to RDF Graphs*. W3C Recommendation 27 October 2009, 2009. http://www.w3.org/TR/owl2-mapping-to-rdf/.

Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.

Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics*, 3:41–60, July 2005.

W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

Harold Boley Gary Hallmark Michael Kifer Adrian Paschke Axel Polleres Dave Reynolds, editor. *OWL 2 Web Ontology Language: Manchester Syntax*. W3C Recommendation 22 June 2010, 2010. http://www.w3.org/TR/rif-core.

S. Rudolph, M. Krötzsch, and P. Hitzler. Cheap Boolean Role Constructors for Description Logics. In S. Hölldobler et al., editors, *Proc. of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08)*, volume 5293 of *Lecture Notes in Artificial Intelligence*, pages 362–374. Springer-Verlag, 2008.