Language Extensions

WonderWeb: Ontology Infrastructure for the Semantic Web

Sean Bechhofer & Ian Horrocks & Jeff Pan
University of Manchester
Kilburn Building
Oxford Road
Manchester M13 9PL
email: {seanb|horrocks|panz}@cs.man.ac.uk

Identifier	Del 3
Class	Deliverable
Version	1.0
Date	30-06-2004
Status	Final
Distribution	Public
Lead Partner	VUM

WonderWeb Project

This document forms part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2001-33052.

For further information about WonderWeb, please contact the project co-ordinator:

Ian Horrocks
The Victoria University of Manchester
Department of Computer Science
Kilburn Building
Oxford Road
Manchester M13 9PL
Tel: +44 161 275 6154
Fax: +44 161 275 6236
Email: wonderweb-info@lists.man.ac.uk

Contents

Administrative Details **1**

1 Introduction **2**

2 A Horn Rules Extension to OWL **3**
2.1 Background and Motivation . . . 3
2.2 Overview . . . 4
2.3 Abstract Syntax . . . 4
2.3.1 Rules . . . 5
2.3.2 Human Readable Syntax . . . 6
2.4 Direct Model-Theoretic Semantics . . . 6
2.4.1 Interpreting Rules . . . 7
2.4.2 Example . . . 7
2.5 XML Concrete Syntax . . . 8
2.5.1 Example . . . 10
2.6 The Power of Rules . . . 10
2.7 Examples of ORL . . . 12
2.7.1 Transferring Characteristics . . . 12
2.7.2 Inferring the Existence of New Individuals . . . 14
2.8 Mapping to RDF Graphs . . . 15
2.9 Reasoning Support for ORL . . . 17
2.10 Summary . . . 18

3 Extending OWL with Complex Role Inclusion Axioms **19**
3.1 Motivation . . . 19
3.2 Preliminaries . . . 21
3.2.1 Relationship with Grammar Logics . . . 24
3.2.2 Role value maps . . . 25
3.3 $\mathcal{SH}^+\mathcal{IQ}$ is undecidable . . . 25
3.4 $\mathcal{RIQ}$ is decidable . . . 28
3.4.1 Translating RIAs into automata . . . 30
3.4.2 A Tableau for $\mathcal{RIQ}$ . . . 33
3.4.3 The Tableau Algorithm . . . 36
3.4.4 Avoiding the blow-up . . . 40
3.5 Evaluation of the $\mathcal{RIQ}$ algorithm in FaCT . . . 40
3.6 Summary and Outlook . . . 42

4 A Datatype Predicate Extension to OWL **43**
4.1 Background and Motivation . . . 43
4.1.1 Datatype Groups . . . 44
4.1.2 Summary . . . 48
4.2 SWRL-P: Extending SWRL with Predicates . . . 48
4.2.1 Abstract Syntax . . . 49
4.2.2 Direct Model Theoretic Semantics . . . 49

4.2.3 SWRL-P vs. SWRL 0.7 50

5 Other Proposed Extensions **52**
5.1 Alternative Semantics for OWL Rules 52
5.2 A Fuzzy Extension to OWL 53
5.3 A Context Extension to OWL 53

6 Conclusion **54**

Executive Summary

The importance of ontologies in the Semantic Web has prompted the development of several proposed extensions to the OWL ontology language. These include extensions for rules, fuzzy concepts, datatypes and multiple contexts. In this report we will study several of the more prominent proposals in detail, and also give an overview of a range of other work in this area.

1 Introduction

Given that ontologies are set to play a key role in the Semantic Web, it is reasonable that one of the first tasks to be tackled in the development of Semantic Web infrastructure was the development and standardisation of a suitable ontology language. This task was finally completed on February 10th 2004 with the recommendation by W3C of the OWL Web Ontology Language [71].

Like its predecessors OIL [22] and DAML+OIL [38], on which it was based, the OWL language is closely related to an expressive Description Logic (DL), in this case $\mathcal{SHOIN}(\mathbf{D}_n)$ [35]. The decision to base all of these languages on DLs was motivated by the many advantages that derive therefrom:[1] the languages come with a well defined semantics; their formal properties are well understood, in particular with respect to the decidability and complexity of key reasoning tasks; sound and complete algorithms for performing these reasoning tasks are well known; and reasoning systems using highly optimised implementations of these algorithms are already available [38, 39].

Another important influence on the design of OWL was the decision to layer the language on top of RDF [6], and to exploit as much as possible of the existing RDF infrastructure. Of particular significance in this regard was the decision to rely on RDF Datatypes, which are themselves based on XML Schema datatypes [10].

Basing the design on DLs and on RDF conferred important advantages on OWL. These advantages do not, however, come without cost: retaining the decidability of reasoning requires the expressive power of the "abstract" part of the language to be constrained, and relying on RDF for datatypes results in the "concrete" part of the language (i.e., the datatypes) being very weak [55]. While acceptable in many contexts, this lack of expressive power can be problematical in many applications, for example in describing web services, where it may be necessary to relate inputs and outputs of composite processes to the inputs and outputs of their component processes [73], or in medical informatics, where it may be necessary to transfer characteristics across partitive properties (an injury to part of an anatomical structure may be considered to be an injury of the structure as a whole) [63]. More expressive datatypes may also be required in many applications, e.g., to constrain values to be in an integer sub-range (the value of age may be constrained to be in the range 0 to 150) or to use higher arity datatype predicates to constrain relationships between multiple values (the value of income may be constrained to be greater than the value of expenditure).

The recognition of these and other limitations has motivated research on several possible extensions to OWL. In this document we will describe three of the most well formed proposals in detail: the OWL Rules Language (ORL) [36], complex role inclusion axioms [40], and datatype predicates [53]. We will also give pointers to other research on extensions to OWL, including alternative rules proposals [20, 23], a "fuzzy" extension to OWL [77], a context extension to OWL (C-OWL) [13] and a proposal for an OWL query language.

[1] See `http://lists.w3.org/Archives/Public/www-webont-wg/`

2 A Horn Rules Extension to OWL

2.1 Background and Motivation

Many of the limitations of OWL stem from the fact that, while the language includes a relatively rich set of class constructors, the language provided for talking about properties is much weaker. In particular, there is no composition constructor, so it is impossible to capture relationships between a composite property and another (possibly composite) property. The standard example here is the obvious relationship between the composition of the "parent" and "brother" properties and the "uncle" property, i.e., we would like to assert that the composition of parent and brother implies uncle.

One way to address this problem would be to extend the DL underlying OWL with a more powerful language for describing properties: in Section 3 we will examine in detail a recent proposal for such an extension. In order to maintain decidability, however, the usage of composition must be limited, and this means that not all relationships between composed properties can be captured—in fact even the relatively simple "uncle" example cannot not be captured (because "uncle" is not one of "parent" or "brother").

An alternative way to overcome some of the expressive restrictions of OWL would be to extend it with some form of "rules language". In fact adding rules to description logic based knowledge representation languages is far from being a new idea. Several early description logic systems, e.g., Classic [59, 12], included a rule language component. In these systems, however, rules were given a weaker semantic treatment than axioms asserting sub- and super-class relationships; they were only applied to individuals, and did not affect class based inferences such as the computation of the class hierarchy. More recently, the CARIN system integrated rules with a description logic in such a way that sound and complete reasoning was still possible [47]. This could only be achieved, however, by using a rather weak description logic (*much* weaker than OWL), and by placing severe syntactic restrictions on the occurrence of description logic terms in the (heads of) rules. Similarly, the DLP language proposed in [24] is based on the intersection of a description logic with horn clause rules; the result is obviously a decidable language, but one that is necessarily less expressive than either the description logic or rules language from which it is formed.

In this section we will present a proposal for an OWL Rules Language (ORL) which adds a simple form of Horn-style rules to OWL. In ORL, rules are syntactically and semantically coherent with the ontology language, the basic idea being to add Horn rules as a new kind of axiom in OWL DL with similar semantics to OWL subClassOf axioms. This proposal has recently been adopted by the DAML Joint Committee on Agent Markup Languages (part of the DARPA DAML programme), renamed the Semantic Web Rules Language (SWRL) and published as a W3C note [37]. As such, it will form an important input to a new W3C "Semantic Web Rules" working group that is likely to be established within the next twelve months.

2.2 Overview

The basic idea of the proposal is to extend OWL DL with a form of rules while maintaining maximum backwards compatibility with OWL's existing syntax and semantics. To this end, we add a new kind of axiom to OWL DL, namely Horn clause rules, extending the OWL abstract syntax and the direct model-theoretic semantics for OWL DL [58] to provide a formal semantics and syntax for OWL ontologies including such rules.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The informal meaning of a rule can be read as: whenever (and however) the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the antecedent (body) and consequent (head) of a rule consist of zero or more atoms. Atoms can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values. Atoms are satisfied in extended interpretations (to take care of variables) in the usual model-theoretic way, i.e., the extended interpretation maps the variables to domain elements in a way that satisfies the description, property, sameAs, or differentFrom, just as in the regular OWL model theory.

Multiple atoms in an antecedent are treated as a conjunction. An empty antecedent is thus treated as trivially true (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation.

Multiple atoms in a consequent are treated as separate consequences, i.e., they must all be satisfied. In keeping with the usual treatment in rules, an empty consequent is treated as trivially false (i.e., not satisfied by any extended interpretation). Such rules are satisfied if and only if the antecedent is not satisfied by any extended interpretation. Note that rules with multiple atoms in the consequent could easily be transformed (via the Lloyd-Topor transformations [48]) into multiple rules each with an atomic consequent.

It is easy to see that OWL DL becomes undecidable when extended in this way as rules can be used to simulate role value maps [69] and make it easy to encode known undecidable problems as an ORL ontology consistency problem (see Section 2.6).

2.3 Abstract Syntax

The syntax for ORL in this section abstracts from any exchange syntax for OWL and thus facilitates access to and evaluation of the language. This syntax extends the abstract syntax of OWL described in the OWL Semantics and Abstract Syntax document [58].

Like the OWL abstract syntax, we will specify the abstract syntax for rules by means of a version of Extended BNF, very similar to the Extended BNF notation used for XML [15]. In this notation, terminals are quoted; non-terminals are not quoted. Alternatives are either separated by vertical bars (|) or are given in different productions. Components that can occur at most once are enclosed in square brackets ([...]); components that can occur any number of times (including zero) are enclosed in braces ({...}). Whitespace is ignored in the productions given here.

Names in the abstract syntax are RDF URI references [43]. These names may be abbreviated into qualified names, using one of the following namespace names:

rdf http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs http://www.w3.org/2000/01/rdf-schema#
xsd http://www.w3.org/2001/XMLSchema#
owl http://www.w3.org/2002/07/owl#

The meaning of each construct in the abstract syntax for rules is informally described when it is introduced. The formal meaning of these constructs is given in Section 2.4 via an extension of the OWL DL model-theoretic semantics [58].

2.3.1 Rules

From the OWL Semantics and Abstract Syntax document [58], an OWL ontology in the abstract syntax contains a sequence of annotations, axioms, and facts. Axioms may be of various kinds, for example, subClass axioms and equivalentClass axioms. This proposal extends axioms to also allow rule axioms, by adding the production:

axiom ::= rule

Thus an ORL ontology could contain a mixture of rules and other OWL DL constructs, including ontology annotations, axioms about classes and properties, and facts about OWL individuals, as well as the rules themselves.

A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms. Just as for class and property axioms, rule axioms can also have annotations. These annotations can be used for several purposes, including giving a label to the rule by using the rdf:label annotation property.

rule ::= 'Implies('{annotation} antecedent consequent')'
antecedent ::= 'Antecedent('{atom}')'
consequent ::= 'Consequent('{atom}')'

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false). Non-empty antecedents and consequents hold iff all of their constituent atoms hold. As mentioned above, rules with multiple consequents could easily transformed (via the Lloyd-Topor transformations [48] into multiple rules each with a single atomic consequent.

Atoms in rules can be of the form C(x), P(x,y), Q(x,z), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property x,y are either variables or OWL individuals, and z is either a variable or an OWL data value. In the context of OWL Lite, descriptions in atoms of the form C(x) may be restricted to class names.

atom ::= description '(' i-object ')'
| individualvaluedPropertyID '(' i-object i-object ')'
| datavaluedPropertyID '(' i-object d-object ')'
| sameAs '(' i-object i-object ')'
| differentFrom '(' i-object i-object ')'

Informally, an atom C(x) holds if x is an instance of the class description C, an atom P(x,y) (resp. Q(x,z)) holds if x is related to y (z) by property P (Q), an atom sameAs(x,y)

holds if x is interpreted as the same object as y, and an atom differentFrom(x,y) holds if x and y are interpreted as different objects.

Atoms may refer to individuals, data literals, individual variables or data variables. Variables are treated as universally quantified, with their scope limited to a given rule. As usual, only variables that occur in the antecedent of a rule may occur in the consequent (a condition usually referred to as "safety"). As we will see in Section 2.6, this safety condition does not, in fact, restrict the expressive power of the language (because existentials can already be captured using OWL someValuesFrom restrictions).

i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral

i-variable ::= 'I-variable(' URIreference ')'
d-variable ::= 'D-variable(' URIreference ')'

2.3.2 Human Readable Syntax

While the abstract Extended BNF syntax is consistent with the OWL specification, and is useful for defining XML and RDF serialisations, it is rather verbose and not particularly easy to read. In the following we will, therefore, often use a relatively informal "human readable" form similar to that used in many published works on rules.

In this syntax, a rule has the form:

$$\mathsf{antecedent} \rightarrow \mathsf{consequent},$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge \ldots \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., $?x$). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

$$\mathsf{parent}(?a, ?b) \wedge \mathsf{brother}(?b, ?c) \rightarrow \mathsf{uncle}(?a, ?c). \tag{1}$$

If John has Mary as a parent and Mary has Bill has a brother, then this rule requires that John has Bill as an uncle.

2.4 Direct Model-Theoretic Semantics

The model-theoretic semantics for ORL is a straightforward extension of the semantics for OWL DL given in [58]. The basic idea is that we define *bindings*—extensions of OWL interpretations that also map variables to elements of the domain in the usual manner. A rule is satisfied by an interpretation iff every binding that satisfies the antecedent also satisfies the consequent. The semantic conditions relating to axioms and ontologies are unchanged, so an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology.

2.4.1 Interpreting Rules

From the OWL Semantics and Abstract Syntax document [58] we recall that an abstract OWL interpretation is a tuple of the form

$$I = \langle R, EC, ER, L, S, LV \rangle,$$

where R is a set of resources, $LV \subseteq R$ is a set of literal values, EC is a mapping from classes and datatypes to subsets of R and LV respectively, ER is a mapping from properties to binary relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to elements of $EC(\texttt{owl:Thing})$.

Given an abstract OWL interpretation I, a binding $B(I)$ is an abstract OWL interpretation that extends I such that S maps i-variables to elements of $EC(\texttt{owl:Thing})$ and L maps d-variables to elements of LV respectively. An atom is satisfied by a binding $B(I)$ under the conditions given in Table 1, where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property, x, y are variables or OWL individuals, and z is a variable or an OWL data value.

Atom	Condition on Interpretation
$C(x)$	$S(x) \in EC(C)$
$P(x,y)$	$\langle S(x), S(y) \rangle \in ER(P)$
$Q(x,z)$	$\langle S(x), L(z) \rangle \in ER(Q)$
sameAs(x,y)	$S(x) = S(y)$
differentFrom(x,y)	$S(x) \neq S(y)$

Table 1: Interpretation Conditions

A binding $B(I)$ satisfies an antecedent A iff A is empty or $B(I)$ satisfies every atom in A. A binding $B(I)$ satisfies a consequent C iff C is not empty and $B(I)$ satisfies every atom in C. A rule is satisfied by an interpretation I iff for every binding B such that $B(I)$ satisfies the antecedent, $B(I)$ also satisfies the consequent.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology O_2 is entailed by an ontology O_1 iff every interpretation that satisfies O_1 also satisfies O_2.

2.4.2 Example

Consider, for example, the "uncle" rule (1) from Section 2.3.2. Assuming that parent, brother and uncle are *individualvaluedPropertyID*s, then given an interpretation $I = \langle R, EC, ER, L, S, LV \rangle$, a binding $B(I)$ extends S to map the variables ?a, ?b, and ?c to elements of $EC(\texttt{owl:Thing})$; we will use a, b, and c respectively to denote these elements. The antecedent of the rule is satisfied by $B(I)$ iff $(a,b) \in ER(\mathsf{parent})$ and $(b,c) \in ER(\mathsf{brother})$. The consequent of the rule is satisfied by $B(I)$ iff $(a,c) \in ER(\mathsf{uncle})$. Thus

the rule is satisfied by I iff for every binding $B(I)$ such that $(a,b) \in ER(\mathsf{parent})$ and $(b,c) \in ER(\mathsf{brother})$, then it is also the case that $(a,c) \in ER(\mathsf{uncle})$, i.e.:

$$\begin{array}{l} \forall a,b,c \in EC(\texttt{owl:Thing}). \\ \quad ((a,b) \in ER(\mathsf{parent}) \wedge (b,c) \in ER(\mathsf{brother})) \\ \quad \rightarrow (a,c) \in ER(\mathsf{uncle}) \end{array}$$

2.5 XML Concrete Syntax

Many possible XML encodings could be imagined (e.g., a RuleML based syntax as proposed in `http://www.daml.org/listarchive/joint-committee/1460.html`), but the most obvious solution is to extend the existing OWL Web Ontology Language XML Presentation Syntax [29], which can be straightforwardly modified to deal with ORL. This has several advantages:

- arbitrary OWL classes (e.g., descriptions) can be used as predicates in rules;
- rules and ontology axioms can be freely mixed;
- the existing XSLT stylesheet[2] can easily be extended to provide a mapping to RDF graphs that extends the OWL RDF/XML exchange syntax (see Section 2.8).

In the first place, the ontology root element is extended so that ontologies can include rule axioms and variable declarations as well as OWL axioms, import statements etc. We then simply need to add the relevant syntax for variables and rules. (In this document we use the unspecified `owlr` namespace prefix. This prefix would have to be bound to some appropriate namespace name, either the OWL namespace name or some new namespace name.)

Variable declarations are statements about variables, indicating that the given URI is to be used as a variable, and (optionally) adding any annotations. For example:

```
<owlr:Variable owlr:name="x1" />,
```

states that the URI `x1` (in the current namespace) is to be treated as a variable.

Rule axioms are similar to OWL `SubClassOf` axioms, except they have `owlr:Rule` as their element name. Like `SubClassOf` and other axioms they may include annotations. Rule axioms have an antecedent (`owlr:antecedent`) component and a consequent (`owlr:consequent`) component. The antecedent and consequent of a rule are both lists of atoms and are read as the conjunction of the component atoms. Atoms can be formed from unary predicates (classes), binary predicates (properties), equalities or inequalities.

Class atoms consist of a description and either an individual name or a variable name, where the description in a class atom may be a class name, or may be a complex description using boolean combinations, restrictions, etc. For example,

```
<owlr:classAtom>
  <owlx:Class owlx:name="Person" />
  <owlr:Variable owlr:name="x1" />
</owlr:classAtom>
```

[2] `http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl`

is a class atom using a class name (#Person), and

```
<owlr:classAtom>
  <owlx:IntersectionOf>
    <owlx:Class owlx:name="Person" />
    <owlx:ObjectRestriction
        owlx:property="hasParent">
      <owlx:someValuesFrom
        owlx:property="Physician" />
    </owlx:ObjectRestriction>
  </owlx:IntersectionOf>
  <owlr:Variable owlr:name="x2" />
</owlr:classAtom>
```

is a class atom using a complex description representing Persons having at least one parent who is a Physician.

Property atoms consist of a property name and two elements that can be individual names, variable names or data values (as OWL does not support complex property descriptions, a property atom takes only a property name). Note that in the case where the second element is an individual name the property must be an *individual-valued* Property, and in the case where the second element is a data value the property must be a *data-valued* Property. For example:

```
<owlr:individualPropertyAtom
                owlx:property="hasParent">
  <owlr:Variable owlr:name="x1" />
  <owlx:Individual owlx:name="John" />
</owlr:individualPropertyAtom>
```

is a property atom using an *individual-valued* Property (the second element is an individual), and

```
<owlr:datavaluedPropertyAtom owlr:property="grade">
  <owlr:Variable owlr:name="x1" />
  <owlx:DataValue
    rdf:datatype="&xsd;integer">4</owlx:DataValue>
</owlr:datavaluedPropertyAtom>
```

is a property atom using a *data-valued* Property datavalued property (the second element is a data value, in this case an integer).

Finally, same (different) individual atoms assert equality (inequality) between sets of individual and variable names. Note that (in)equalities can be asserted between arbitrary combinations of variable names and individual names. For example:

```
<owlr:sameIndividualAtom>
  <owlr:Variable owlr:name="x1" />
  <owlr:Variable owlr:name="x2" />
```

```
  <owlx:Individual owlx:name="Clinton" />
  <owlx:Individual owlx:name="Bill_Clinton" />
</owlr:sameIndividualAtom>
```

asserts that the variables $x1$, $x2$ and the individual names Clinton and Bill_Clinton all refer to the same individual.

2.5.1 Example

The example rule from Section 2.3.2 can be written in the XML concrete syntax for rules as

```
<owlx:Rule>
  <owlr:antecedent>
    <owlr:individualPropertyAtom
                    owlr:property="parent">
      <owlr:Variable owlr:name="a" />
      <owlr:Variable owlr:name="b" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom
                    owlr:property="brother">
      <owlr:Variable owlr:name="b" />
      <owlr:Variable owlr:name="c" />
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent>
    <owlr:individualPropertyAtom
                    owlr:property="uncle">
      <owlr:Variable owlr:name="a" />
      <owlr:Variable owlr:name="c" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

2.6 The Power of Rules

In OWL, the only relationship that can be asserted between properties is subsumption between atomic property names, e.g., asserting that hasFather is a subPropertyOf hasParent. In Section 2.3.2 we have already seen how a rule can be used to assert more complex relationships between properties. While this increased expressive power is clearly very useful, it is easy to show that it leads to the undecidability of key inference problems, in particular ontology consistency.

For extensions of languages such as OWL DL, the undecidability of the consistency problem is often proved by showing that the extension makes it possible to encode a known undecidable domino problem [7] as an ontology consistency problem. In particular, it is well known that such languages only need the ability to represent an infinite

2-dimensional grid in order for consistency to become undecidable [3, 41]. With the addition of rules, such an encoding is trivial. For example, given two properties x-succ and y-succ, the rule:

$$\begin{array}{ll} \mathsf{x\text{-}succ}(?a,?b) \wedge \mathsf{y\text{-}succ}(?b,?c) \wedge & \\ \mathsf{y\text{-}succ}(?a,?d) \wedge \mathsf{x\text{-}succ}(?d,?e) & \rightarrow \quad \mathtt{sameAs}(?c,?e), \end{array}$$

along with the assertion that every grid node is related to exactly one other node by each of x-succ and y-succ, allows such a grid to be represented. This would be possible even without the use of the sameAs atom in the consequent—it would only be necessary to establish appropriate relationships with a "diagonal" property:

$$\begin{array}{lll} \mathsf{x\text{-}succ}(?a,?b) \wedge \mathsf{y\text{-}succ}(?b,?c) & \rightarrow & \mathsf{diagonal}(?a,?c) \\ \mathsf{y\text{-}succ}(?a,?d) \wedge \mathsf{x\text{-}succ}(?d,?e) & \rightarrow & \mathsf{diagonal}(?a,?e), \end{array}$$

and additionally assert that every grid node is related to exactly one other node by diagonal.

The proposed form of OWL rules seem to go beyond basic Horn clauses in allowing:

- conjunctive consequents;
- class descriptions as well as class names as predicates in class atoms; and
- equalities and inequalities.

On closer examination, however, it becomes clear that most of this is simply "syntactic sugar", and does not add to the power of the language.

In the case of conjunctive consequents, it is easy to see that these could be eliminated using the standard Lloyd-Topor transformation [48]. For example, a rule of the form

$$A \rightarrow C_1 \wedge C_2$$

can be transformed into a semantically equivalent pair of rules

$$\begin{array}{l} A \rightarrow C_1 \\ A \rightarrow C_2. \end{array}$$

In the case of class descriptions, it is easy to see that a description d can be eliminated from a rule simply by adding an OWL axiom that introduces a new class name and asserts that it is equivalent to d, e.g.,

$$\mathtt{EquivalentClasses}(D\ d).$$

The description can then be replaced with the name, here replacing the description d with class name D.

In the case of equality atoms, the sameAs predicate could easily be substituted with a "user defined" owl property called, for example, Eq. Such a property can be given the appropriate meaning using a rule of the form

$$\mathtt{Thing}(?x) \rightarrow \mathsf{Eq}(?x,?x)$$

and by asserting that it is functional.

The case of inequalities is slightly more complex. When they occur in the consequent of a rule they can easily be eliminated. For example, the atom

$$\texttt{differentFrom}(x,y),$$

where x and y are again variables or constants, can be replaced with

$$C(x) \wedge D(y),$$

where C and D are new class names that are asserted to be disjoint. When (in)equalities occur in antecedents, however, this elimination does not work, because it would strengthen the conditions that must be met in order for a binding to satisfy the antecedent.

2.7 Examples of ORL

We give two further examples of ORL that serve to illustrate some of their utility, and show how the power of ORL goes beyond that of either OWL DL or Horn rules alone.

2.7.1 Transferring Characteristics

The first example is due to Guus Schreiber, and is based on ontologies used in an image annotation demo [27].

$$\begin{aligned}&\mathsf{Artist}(?x) \wedge \mathsf{Style}(?y) \wedge \mathsf{artistStyle}(?x,?y) \wedge \mathsf{creator}(?x,?z)\\ &\rightarrow \mathsf{style/period}(?z,?y)\end{aligned}$$

The rule expresses the fact that, given knowledge about the Style of certain Artists (e.g., van Gogh is an Impressionist painter), we can derive the style/period of an art object from the value of the creator of the art object, where Style is a term from the Art and Architecture Thesaurus (AAT),[3] Artist is a class from the Union List of Artist Names (ULAN),[4] artistStyle is a property relating ULAN Artists to AAT Styles, and both creator and style/period are properties from the Visual Resources Association catalogue (VRA),[5] with creator being a subproperty of the Dublin Core element dc:creator.[6]

This rule would be expressed in the XML concrete syntax as follows (assuming appropriate entity declarations):

```
<owlr:Rule>
  <owlr:antecedent>
    <owlr:classAtom>
      <owlx:Class owlx:name="&ulan;Artist" />
      <owlr:Variable owlr:name="x" />
    </owlr:classAtom>
```

[3] http://www.getty.edu/research/tools/vocabulary/aat/
[4] http://www.getty.edu/research/conducting_research/vocabularies/ulan/
[5] http://www.vraweb.org/
[6] http://dublincore.org/

```
    <owlr:classAtom>
      <owlx:Class owlx:name="&aat;Style" />
      <owlr:Variable owlr:name="y" />
    </owlr:classAtom>
    <owlr:individualPropertyAtom
            owlr:property="&aatulan;artistStyle">
      <owlr:Variable owlr:name="x" />
      <owlr:Variable owlr:name="y" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom
            owlr:property="&vra;creator">
      <owlr:Variable owlr:name="x" />
      <owlr:Variable owlr:name="z" />
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent>
    <owlr:individualPropertyAtom
            owlr:property="&vra;style/period">
      <owlr:Variable owlr:name="z" />
      <owlr:Variable owlr:name="y" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

The example is interesting because it shows how rules can be used to "transfer characteristics" from one class of individuals to another via properties other than subClassOf—in this case, the Style characteristics of an Artist (if any) are transferred (via the creator property) to the objects that he/she creates. This idiom is much used in ontologies describing complex physical systems, such as medical terminologies, where partonomies may be as important as subsumption hierarchies, and where characteristics often need to be transfered across various partitive properties [52, 62, 66]. For example, the location of a trauma should be transfered across the partOf property, so that traumas located in a partOf an anatomical structure are also located in the structure itself [63]. This could be expressed using a rule such as

$$\begin{aligned}&\mathsf{Location}(?x) \wedge \mathsf{Trauma}(?y) \wedge \mathsf{isLocationOf}(?x, ?y) \wedge \\ &\ \mathsf{isPartOf}(?x, ?z) \\ &\rightarrow \mathsf{isLocationOf}(?z, ?y)\end{aligned}$$

A similar technique could be used to transfer properties to composite processes from their component processes when describing web services.

Terminology languages designed specifically for medical terminology such as Grail [61] and SNOMED-RT [72] often allow this kind of idiom to be expressed, but it cannot be expressed in OWL (not even in OWL full). Thus this kind of rule shows one way in which ORL go beyond the expressive power of OWL DL.

2.7.2 Inferring the Existence of New Individuals

The second example is due to Mike Dean, and illustrates a scenario in which we want to express the fact that for every Airport there is a map Point that has the same location (latitude and longitude) as the Airport and that is an object of "layer" (a map DrawingLayer).[7] Moreover, this map point has the Airport as an underlyingObject and has the Airport name as its Label. Note how the expressive power of ORL allows "existentials" to be expressed in the head of a rule—it is asserted that, for every airport, there must exist such a map point (using an OWL someValuesFrom restriction in a class atom). In this way ORL goes beyond the expressive power of Horn rules.

The first part of this example is background knowledge about airports and maps expressed in OWL DL. (A few liberties have been taken with the OWL DL abstract syntax here in the interests of better readability.) In particular, it is stated that map:location and map:object are *individual-valued* Properties with inverse properties map:isLocationOf and map:isObjectOf respectively; that latitude and longitude are *data-valued* Properties; that map:Location is a class whose instances have exactly one latitude and exactly one longitude, both being of type xsd:double; that layer is an instance of map:DrawingLayer; that map is an instance of map:Map whose map:name is `"Airports"` and whose map:layer is layer; and that airport:GEC is an instance of airport-ont:airport whose name is `"Spokane Intl"` and whose location is latitude `47.6197` and longitude `117.5336`.

```
ObjectProperty(map:location)
ObjectProperty(map:isLocationOf
  inverseOf(map:location))
ObjectProperty(map:object)
ObjectProperty(map:isObjectOf
  inverseOf(map:location))

DatatypeProperty(latitude)
DatatypeProperty(longitude)
Class(map:Location primitive
  intersectionOf(
    restriction(latitude allValuesFrom(xsd:double))
    restriction(latitude minCardinality(1))
    restriction(longitude allValuesFrom(xsd:double))
    restriction(longitude minCardinality(1))))

Individual(layer type(map:DrawingLayer))

Individual(map type(map:Map)
  value(map:name "Airports)
  value(map:layer layer))

Individual(airport:GEC type(airport-ont:airport)
```

[7] `http://www.daml.org/2003/06/ruletests/translation-3.n3`

```
value(name "Spokane Intl")
value(location Individual(value(latitude 47.6197)
      value(longitude 117.5336))))
```

The first rule in the example requires that if a `map:Location` is the `sameLocation` as another `location`, then it has the same values for `latitude` and `longitude`.

$$\begin{array}{l}\text{map:Location}(?maploc) \wedge \text{sameLocation}(?loc, ?maploc) \wedge \\ \text{latitude}(?loc, ?lat) \wedge \text{longitude}(?loc, ?lon) \\ \rightarrow \\ \text{latitude}(?maploc, ?lat) \wedge \text{latitude}(?maploc, ?lon)\end{array}$$

The second rule requires that wherever an `airport-ont:Airport` is located, there is some `map:Location` that is the `sameLocation` as the airport's location, and that is the location of a `map:Point` that is an object of the `map:DrawingLayer` "`layer`".

$$\begin{array}{l}\text{airport-ont:Airport}(?airport) \wedge \text{location}(?airport, ?loc) \wedge \\ \text{latitude}(?loc, ?lat) \wedge \text{longitude}(?loc, ?lon) \\ \rightarrow \\ \texttt{restriction}(\text{sameLocation} \\ \texttt{someValuesFrom}(\\ \texttt{intersectionOf}(\text{map : Location} \\ \texttt{restriction}(\text{isLocationOf} \\ \texttt{someValuesFrom}(\\ \texttt{intersectionOf}(\text{map : Point} \\ \texttt{restriction}(\text{map : isObjectOf} \\ \texttt{someValuesFrom}(\texttt{OneOf}(\text{layer}))))))))) \\ (?loc)\end{array}$$

The third rule requires that the `map:Point` whose `map:location` is the `map:Location` of an `airport-ont:Airport` has the airport as a `map:underlyingObject` and has a `map:label` which is the name of the airport.

$$\begin{array}{l}\text{airport-ont:Airport}(?airport) \wedge \\ \text{map:location}(?airport, ?loc) \wedge \\ \text{sameLocation}(?loc, ?maploc) \wedge \\ \text{map:Location}(?point, ?maploc) \wedge \\ \text{airport-ont:name}(?airport, ?name) \\ \rightarrow \\ \text{map:underlyingObject}(?point, ?airport) \wedge \\ \text{map:label}(?point, ?name)\end{array}$$

2.8 Mapping to RDF Graphs

It is widely assumed that the Semantic Web will be based on a hierarchy of (increasingly expressive) languages, with RDF/XML providing the syntactic and semantic foundation (see, e.g., [9]). In accordance with this design philosophy, the charter of the W3C Web

Ontology Working Group (the developers of the OWL language) explicitly stated that "*The language will use the XML syntax and datatypes wherever possible, and will be designed for maximum compatibility with XML and RDF language conventions.*". In pursuance of this goal, the working group devoted a great deal of effort to developing an RDF based syntax for OWL that was also consistent with the semantics of RDF [39]. It is, therefore, worth considering how this design might be extended to encompass rules.

One rather serious problem is that, unlike OWL, rules have variables, so treating them as a semantic extension of RDF is very difficult. It is, however, still possible to provide an RDF syntax for rules—it is just that the semantics of the resultant RDF graphs may not be an extension of the RDF Semantics [26].

A mapping to RDF/XML is most easily created as an extension to the XSLT transformation for the OWL XML Presentation syntax.[8] This would introduce RDF classes for ORL atoms and variables, and RDF properties to link atoms to their predicates (classes and properties) and arguments (variables, individuals or data values).[9] The example rule given in Section 2.7.1 (that equates the style/period of art objects with the style of the artist that created them) would be mapped into RDF as follows:

```
<owlr:Variable rdf:ID="x"/>
<owlr:Variable rdf:ID="y"/>
<owlr:Variable rdf:ID="z"/>
<owlr:Rule>
  <owlr:antecedent rdf:parseType="Collection">
    <owlr:classAtom>
      <owlr:classPredicate
        rdf:resource="&ulan;Artist"/>
      <owlr:argument1 rdf:resource="#x" />
    </owlr:classAtom>
    <owlr:classAtom>
      <owlr:classPredicate
        rdf:resource="&aat;Style"/>
      <owlr:argument1 rdf:resource="#y" />
    </owlr:classAtom>
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&aatulan;artistStyle"/>
      <owlr:argument1 rdf:resource="#x" />
      <owlr:argument2 rdf:resource="#y" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&vra;creator"/>
      <owlr:argument1 rdf:resource="#x" />
      <owlr:argument2 rdf:resource="#z" />
```

[8] http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl

[9] The result is similar to the RDF syntax for representing disjunction and quantifiers proposed in [50].

```
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent rdf:parseType="Collection">
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate
        rdf:resource="&vra;style/period"/>
      <owlr:argument1 rdf:resource="#z" />
      <owlr:argument2 rdf:resource="#y" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

where `&ulan;`, `&aat;`, `&aatulan;` and `&vra;` are assumed to expand into the appropriate namespace names. Note that complex OWL classes (such as OWL restrictions) as well as class names can be used as the object of ORL's classPredicate property.

2.9 Reasoning Support for ORL

Although ORL provides a fairly minimal rule extension to OWL, the consistency problem for ORL ontologies is still undecidable (as we have seen in Section 2.6). This raises the question of how reasoning support for ORL might be provided.

It seems likely, at least in the first instance, that many implementations will provide only partial support for ORL. For this reason, users may want to restrict the form or expressiveness of the rules and/or axioms they employ either to fit within a tractable or decidable fragment of ORL, or so that their ORL ontologies can be handled by existing or interim implementations.

One possible restriction in the form of the rules is to limit antecedent and consequent classAtoms to be named classes, with OWL axioms being used to assert additional constraints on the instances of these classes (in the same document or in external OWL documents). Adhering to this format should make it easier to translate rules to or from existing (or future) rule systems, including Prolog, production rules (descended from OPS5), event-condition-action rules and SQL (where views, queries, and facts can all be seen as rules); it may also make it easier to extend existing rule based reasoners for OWL (such as Euler[10] or FOWL[11]) to handle ORL ontologies. Further, such a restriction would maximise backwards compatibility with OWL-speaking systems that do not support ORL. It should be pointed out, however, that there may be some incompatibility between the first order semantics of ORL and the Herbrand model semantics of many rule based reasoners.

By further restricting the form of rules and DL axioms used in ORL ontologies it would be possible to stay within DLP, a subset of the language that has been shown to be expressible in either OWL DL or declarative logic programs (LP) alone [24]. This would allow either OWL DL reasoners or LP reasoners to be used with such ontologies, although there may again be some incompatibility between the semantics of ORL and those of LP reasoners.

[10] `http://www.agfa.com/w3c/euler/`
[11] `http://fowl.sourceforge.net`

Another obvious strategy would be to restrict the form of rules and DL axioms so that a "hybrid" system could be used to reason about the resulting ontology. This approach has been used, e.g., in the CLASSIC [59] and CARIN systems [47], where sound and complete reasoning is made possible mainly by focusing on query answering, by restricting the DL axioms to languages that are *much* weaker than OWL, by restricting the use of DL terms in rules, and/or by giving a different semantic treatment to rules.

Finally, an alternative way to provide reasoning support for ORL would be to extend the translation of OWL into TPTP[12] implemented in the Hoolet system,[13] and use a first order prover such as Vampire to reason with the resulting first order theory [64, 76]. This technique would have several advantages: no restrictions on the form of ORL rules or axioms would be required; the use of a first order prover would ensure that all inferences were sound with respect to ORL's first order semantics; and the use of the TPTP syntax would make it possible to use any one of a range of state of the art first order provers.

2.10 Summary

In this section we have presented ORL, a proposed extension to OWL to include a simple form of Horn-style rules. We have provided formal syntax and semantics for ORL, shown how OWL's XML and RDF syntax can be extended to deal with ORL, illustrated the features of ORL with several examples, and discussed how reasoning support for ORL might be provided.

The main strengths of ORL are its simplicity and its tight integration with the existing OWL language. As we have seen, ORL extends owl with the most basic kind of Horn rule (sweetened with a little "syntactic sugar"): predicates are limited to being OWL classes and properties (and so have a maximum arity of 2), there are no disjunctions or negations (of atoms), no built in predicates (such as arithmetic predicates), and no nonmonotonic features such as negation as failure or defaults. Moreover, rules are given a standard first order semantics. This facilitates the tight integration with OWL, with ORL being defined as a syntactic and semantic extension of OWL DL.

[12] A standard syntax used by many first order theorem provers—see `http://www.tptp.org`
[13] `http://www.w3.org/2003/08/owl-systems/test-results-out`

3 Extending OWL with Complex Role Inclusion Axioms

3.1 Motivation

As we have already discussed, the OWL language is based on the $\mathcal{SHOIN}(\mathbf{D}_n)$ DL which is itself closely related to the well known $\mathcal{SHIQ}$ DL. The $\mathcal{SHIQ}$ DL [42, 32] is an expressive knowledge representation formalism that extends $\mathcal{ALC}$ [68] with qualifying number restrictions, inverse roles, role inclusion axioms, and transitive roles. The development of $\mathcal{SHIQ}$ was motivated and inspired by several applications, one of which was the representation of knowledge about complex physically structured domains found, e.g., in chemical engineering [65] and medical terminology [62].

For example, in $\mathcal{SHIQ}$, we can describe fractures of the femur by the following concept which, intuitively, denotes fractures that are located in the femur or the neck of the femur:

$$\texttt{FemurFracture} \doteq \texttt{Fracture} \sqcap \exists \texttt{hasLocation}.(\texttt{Femur} \sqcup \texttt{FemurNeck}).$$

To make this definition work, we also should describe the neck of the femur, e.g., as follows:

$$\texttt{FemurNeck} \doteq \texttt{BodyPart} \sqcap \texttt{Proxima} \exists \texttt{isDivisionOf}.\texttt{Femur}.$$

$\mathcal{SHIQ}$ allows many important properties of application domains to be captured: e.g., we can state that `hasLocation` is transitive, and that `LocatedIn` is the inverse of `hasLocation`. However, there is one extremely useful feature that $\mathcal{SHIQ}$ cannot express, namely the "propagation" of one property along another property [52, 60, 72]. Coming back to our example above, to capture that also a fracture of the shaft of the femur is a fracture of the femur, we need to add this information explicitly the definition of `FemurFracture`. As such, this is easily feasible. A more elegant approach would be to change our definition to

$$\texttt{FemurFracture} \doteq \texttt{Fracture} \sqcap \exists \texttt{hasLocation}.(\texttt{Femur} \sqcup \exists \texttt{isDivisionOf}.\texttt{Femur}).$$

Still, we have to have a similar disjunction in the definition of the fracture of the tibia, and all other fractures. Thus, it would be useful if we could express, in general, the fact that certain locative properties are transfered across certain partonomic properties so that a fracture or trauma located in a part of a body structure is recognised as being located in the body structure as a whole. This would yield the highly desirable inferences such as a fracture of the shaft of the femur being inferred to be a kind of fracture of the femur, or an ulcer located in the gastric mucosa being inferred to be a kind of stomach ulcer—without the necessity to repeat this statement in the definition of every single such concept.

The importance of these kinds of inferences, particularly in medical terminology applications, is illustrated by the fact that three different such applications provide means to express propagation. The Grail DL [61], which was specifically designed for use with medical terminology, is able to represent these kinds of propagation (although it is quite weak in other respects). In another medical terminology application using the comparatively inexpressive DL $\mathcal{ALC}$, a rather complex "work around" is performed in order to represent similar propagations [70]: so-called *SEP-triplets* are used both to compensate

for the absence of transitive roles in $\mathcal{ALC}$, and to express the propagation of properties across a distinguished "part-of" role. In a third application, use is made of so-called *right-identities*, which correspond to our complex role inclusion axioms [72]. Finally, similar expressiveness was also provided in the CycL language by the `transfersThro` statement [46]. To the best of our knowledge, however, there is no proof of the correct treatment of propagation in any of these applications.

As we have seen in Section 2, one way to address this problem is to extend the language with Horn clause rules, but that this immediately leads to the undecidability of key inference problems. An alternative aproach is to extend $\mathcal{SHIQ}$ so that this kind of role propagation can be expressed: simply allow for role inclusion axioms (RIAs) of the form $R \circ S \dot{\sqsubseteq} P$, which then forces all models I to interpret the composition of R^I with S^I as a sub-relation of P^I. E.g., the above examples translate into

$$\texttt{hasLocation} \circ \texttt{isDivisionOf} \dot{\sqsubseteq} \texttt{hasLocation},$$

which implies that

$$\texttt{Fracture} \sqcap \exists \texttt{hasLocation}.(\texttt{Neck} \sqcap \exists \texttt{isDivisionOf}.\texttt{Femur}),$$

i.e., a concept describing fractures of the neck of the femur, is indeed subsumed by (is a specialisation of)

$$\texttt{Fracture} \sqcap \exists \texttt{hasLocation}.\texttt{Femur},$$

i.e., a concept describing fractures of the femur.

Unfortunately, this extension also leads to the undecidability of interesting inference problems such as concept satisfiability and subsumption [78]. This undecidability is not surprising once we observe the close relationship between RIAs, *Grammar Logics* [4, 5, 18], and *role value maps* [14, 69]. This relationship is discussed in more detail in Section 3.2.1. Here, it should suffice to mention that a RIA $RS \dot{\sqsubseteq} T$ can be viewed as a notational variant of the production rule $T \rightarrow RS$ of Grammar Logics or the concept inclusion $\top \dot{\sqsubseteq} (RS \tilde{\sqsubseteq} T)$ of a description logic allowing for role value maps.

On closer inspection of our motivating examples, we observe that only RIAs of the form $RS \dot{\sqsubseteq} S$ or $SR \dot{\sqsubseteq} S$ are required in order to express propagation. To the best of our knowledge, no (un)decidability results are known for similar restrictions of the above mentioned Grammar Logics or DLs with role value maps. In this paper, we will show that $\mathcal{SHIQ}$ extended with this restricted form of RIAs is still undecidable. Due to the syntactic restrictions imposed on RIAs, we cannot re-use techniques employed to prove undecidability of Grammar Logics or DLs with role value maps. Instead, our proof is by reduction of the undecidable domino problem [8], and uses a rather special technique to ensure a grid structure.

Decidability can be regained, however, by further restricting the set of RIAs to be *regular*, and the logic obtained by restricting RIAs to regular ones is called $\mathcal{RIQ}$. From a practical point of view, the restrictions imposed by regularity do not seem to be severe: regular RIAs should suffice for many applications, and non-regular RIAs may even be an indicator of modelling flaws [60].

We prove the decidability of $\mathcal{SHIQ}$ with regular RIAs via a tableau-based decision procedure for the satisfiability of concepts. We first translate regular RIAs into non-deterministic automata, and then use these automata in the tableau algorithm. More pre-

cisely, the tableau algorithm replaces concepts of the form $\forall R.C$ (where R is a role) with expressions of the form $\forall \mathcal{B}_R.C$, where $\mathcal{B}_R$ is a non-deterministic finite automaton (NFA) capturing exactly the restrictions imposed on R by RIAs. Using these expressions, we ensure that the concept C is indeed "pushed" to all those nodes it has to be pushed to, even if they are far away from a node that has to satisfy $\forall R.C$. The algorithm is of the same complexity as the one for $\mathcal{SHIQ}$—in the size of $\mathcal{B}_R$ and the length of the input concept—but, unfortunately, $\mathcal{B}_R$ can be exponential in the "depth" of $\mathcal{R}$, i.e., in the length of chains of roles depending on each other. We also present a syntactic restriction that avoids this blow-up; investigating whether this blow-up can be avoided in general will be part of future work.

As we have discussed above, the interaction between roles in regular RIAs can be captured by NFAs, but we have not yet explained which RIAs are regular. This is so because, in the presence of inverse roles, the definition of regularity becomes slightly tricky: each "left-linear" RIA of the form $RS \dot{\sqsubseteq} S$ is equivalent to a "right-linear" RIA $S^- R^- \dot{\sqsubseteq} S^-$. Thus each left-linear RIA has consequences that are inherently a mixture of right- and left-linear RIAs. Now it is well-known that grammars with a such a linear mixture are stronger than right-linear grammars or left-linear grammars [28], and this is true also for RIAs, as our undecidability result shows. Thus, to enable the transformation into an automaton, we impose an additional restriction, which we have chosen to be *acyclicity* in a rather loose sense, i.e., we still allow for RIAs $SS \dot{\sqsubseteq} S$, $RS \dot{\sqsubseteq} S$, and $SR \dot{\sqsubseteq} S$, but we do not allow for combinations of RIAs such as $RS \dot{\sqsubseteq} S$ and $SR \dot{\sqsubseteq} R$.

Finally, in order to evaluate the practicability of this algorithm, we have extended the DL system FaCT [30] to deal with $\mathcal{RIQ}$. We discuss how the properties of NFAs are exploited in the implementation, and we present some preliminary results showing that the performance of the extended system is comparable with that of the original, and that it is able to compute inferences of the kind mentioned above w.r.t. the well-known Galen medical terminology knowledge base [62, 30].

3.2 Preliminaries

In this section, we introduce the DL $\mathcal{SH}^+\mathcal{IQ}$. This includes the definition of syntax, semantics, and inference problems.

Definition 1 *Let* **C** *be a set of* concept names *and* **R** *a set of* role names. *The set of* roles *is* $\mathbf{R} \cup \{R^- \mid R \in \mathbf{R}\}$. *A* role inclusion axiom *is an expression of one of the following forms:*

$$R_1 \dot{\sqsubseteq} R_2, \quad R_1 R_2 \dot{\sqsubseteq} R_1, \text{ or } R_1 R_2 \dot{\sqsubseteq} R_2,$$

for roles R_i (each of which can be inverse). A generalised role hierarchy *is a set of role inclusion axioms.*

An interpretation $I = (\Delta^I, \cdot^I)$ associates, with each role name R, a binary relation $R^I \subseteq \Delta^I \times \Delta^I$. Inverse roles are interpreted as usual, i.e.,

$$(R^-)^I \quad = \quad \{\langle y, x\rangle \mid \langle x, y\rangle \in R^I\} \quad \text{for each role } R \in \mathbf{R}.$$

An interpretation I is a model *of a generalised role hierarchy $\mathcal{R}$ if it satisfies each inclusion assertion in $\mathcal{R}$, i.e., if*

$$\begin{array}{rcll} R_1^I & \subseteq & R_2^I & \text{for each } R_1 \dot{\sqsubseteq} R_2 \in \mathcal{R} \text{ and} \\ R_1^I \circ R_2^I & \subseteq & R_3^I & \text{for each } R_1 R_2 \dot{\sqsubseteq} R_3 \in \mathcal{R}, \end{array}$$

where $\circ$ stands for the composition of binary relations.

Note that we did not introduce *transitive role names* since adding $RR \dot{\sqsubseteq} R$ to the generalised role hierarchy is equivalent to saying that R is a transitive role.

To avoid considering roles such as R^{--}, we define a function Inv on roles such that $\mathsf{Inv}(R) = R^-$ if R is a role name, and $\mathsf{Inv}(R) = S$ if $R = S^-$.

Since we will often work with a string of roles, it is convenient to extend both $\cdot^I$ and $\mathsf{Inv}(\cdot)$ to such strings: if $w = R_1 \dots R_n$ for R_i roles, then $w^I = R_1^I \circ \dots \circ R_n^I$ and $\mathsf{Inv}(w) = \mathsf{Inv}(R_n) \dots \mathsf{Inv}(R_1)$. It follows immediately from the definition of the semantics that

$$\langle x, y \rangle \in w^I \text{ iff } \langle y, x \rangle \in \mathsf{Inv}(w)^I.$$

Next, since each model satisfying $w \dot{\sqsubseteq} S$ also satisfies $\mathsf{Inv}(w) \dot{\sqsubseteq} \mathsf{Inv}(S)$ (and vice versa), we can restrict generalised role hierarchies to those with role *names* on their right hand side without any effect on the expressivity. For better readability, we will not do this in the undecidability proof of $\mathcal{SH}^+\mathcal{IQ}$, but we will do it for the decidable logic $\mathcal{RIQ}$ since it makes the construction in the proofs easier.

Finally, for a generalised role hierarchy $\mathcal{R}$, we define the relation $\dot{\sqsubseteq}^*$ to be the transitive-reflexive closure of $\dot{\sqsubseteq}$ over $\{R \dot{\sqsubseteq} S, \mathsf{Inv}(R) \dot{\sqsubseteq} \mathsf{Inv}(S) \mid R, S \text{ roles and } R \dot{\sqsubseteq} S \in \mathcal{R}\}$. A role R is called a *sub-role* (resp. super-role) of a role S if $R \dot{\sqsubseteq}^* S$ (resp. $S \dot{\sqsubseteq}^* R$). Two roles R and S are *equivalent* ($R \equiv S$) if $R \dot{\sqsubseteq}^* S$ and $S \dot{\sqsubseteq}^* R$.

Now we are ready to define the syntax and semantics of $\mathcal{SH}^+\mathcal{IQ}$-concepts.

Definition 2 *Let $\mathcal{R}$ be a generalised role hierarchy. A role R is* simple in $\mathcal{R}$ *if, for each $R' \dot{\sqsubseteq}^* R$, $\mathcal{R}$ contains no RIA of the form $R_1 R_2 \dot{\sqsubseteq} R'$ or $R_1 R_2 \dot{\sqsubseteq} \mathsf{Inv}(R')$. If $\mathcal{R}$ is clear from the context, we often use "simple" instead of "simple in $\mathcal{R}$".*

*The set of $\mathcal{SH}^+\mathcal{IQ}$-*concepts *is the smallest set such that*

- *every concept name and $\top, \bot$ are concepts, and,*
- *if C, D are concepts, R is a role (possibly inverse), S is a simple role (possibly inverse) , and n is a non-negative integer, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $(\geqslant nS.C)$, and $(\leqslant nS.C)$ are also concepts.*

A general concept inclusion axiom *(GCI) is an expression of the form $C \dot{\sqsubseteq} D$ for two $\mathcal{SH}^+\mathcal{IQ}$-concepts C and D. A* terminology *is a set of GCIs.*

An interpretation *$I = (\Delta^I, \cdot^I)$ consists of a set Δ^I, called the* domain *of I, and a* valuation *$\cdot^I$ which maps every concept to a subset of Δ^I and every role to a subset of $\Delta^I \times \Delta^I$ such that, for all concepts C, D, roles R, S, and non-negative integers n, the following equations are satisfied, where $\sharp M$ denotes the cardinality of a set M:*

$$
\begin{array}{rll}
\top^I = \Delta^I & \bot^I = \emptyset & \textit{(top and bottom)} \\
(C \sqcap D)^I = C^I \cap D^I & & \textit{(conjunction)} \\
(C \sqcup D)^I = C^I \cup D^I & & \textit{(disjunction)} \\
(\neg C)^I = \Delta^I \setminus C^I & & \textit{(negation)} \\
(\exists R.C)^I = \{x \mid \exists y.\langle x,y\rangle \in R^I \textit{ and } y \in C^I\} & & \textit{(exists restriction)} \\
(\forall R.C)^I = \{x \mid \forall y.\langle x,y\rangle \in R^I \textit{ implies } y \in C^I\} & & \textit{(value restriction)} \\
(\geqslant nR.C)^I = \{x \mid \sharp\{y.\langle x,y\rangle \in R^I \textit{ and } y \in C^I\} \geqslant n\} & & \textit{(at least restriction)} \\
(\leqslant nR.C)^I = \{x \mid \sharp\{y.\langle x,y\rangle \in R^I \textit{ and } y \in C^I\} \leqslant n\} & & \textit{(at most restriction)}
\end{array}
$$

An interpretation I is a model *of a terminology $\mathcal{T}$ (written $I \models \mathcal{T}$) iff $C^I \subseteq D^I$ for each GCI $C \dot{\sqsubseteq} D$ in $\mathcal{T}$.*

A concept C is called satisfiable *iff there is an interpretation I with $C^I \neq \emptyset$. A concept D* subsumes *a concept C (written $C \sqsubseteq D$) iff $C^I \subseteq D^I$ holds for each interpretation. Two concepts are* equivalent *(written $C \equiv D$) if they are mutually subsuming. The above inference problems can be defined w.r.t. a generalised role hierarchy $\mathcal{R}$ and/or a terminology $\mathcal{T}$ in the usual way, i.e., by replacing* interpretation *with* model of $\mathcal{R}$ *and/or* $\mathcal{T}$.

For an interpretation I, an element $x \in \Delta^I$ is called an instance *of a concept C iff $x \in C^I$.*

Please note that number restrictions $(\geqslant nR.C)$ and $(\leqslant nR.C)$ are restricted to *simple* roles. Intuitively, these are (possibly inverse) roles that are not implied by the composition of other roles. The reason for this restriction is that, without it, satisfiability of $\mathcal{SHIQ}$-concepts is undecidable [34], even for a logic without inverse roles and with only *unqualifying* number restrictions (these are number restrictions of the form $(\geqslant nR.\top)$ and $(\leqslant nR.\top)$).

For DLs that are closed under negation, subsumption and (un)satisfiability can be mutually reduced: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable, and C is unsatisfiable iff $C \sqsubseteq \bot$. It is straightforward to extend these reductions to generalised role hierarchies and terminologies. In contrast, the reduction of inference problems w.r.t. a terminology to pure concept inference problems (possibly w.r.t. a role hierarchy), deserves special care: in [1, 67, 2], the *internalisation* of GCIs is introduced, a technique that realises exactly this reduction. For $\mathcal{SH}^+\mathcal{IQ}$, this technique only needs to be slightly modified. The following Lemma shows how general concept inclusion axioms can be *internalised* using a "universal" role U, that is, a transitive super-role of all roles occurring in $\mathcal{T}$ or $\mathcal{R}$ and their respective inverses.

Lemma 3 *Let C, D be concepts, $\mathcal{T}$ a terminology, and $\mathcal{R}$ a generalised role hierarchy. We define*

$$C_{\mathcal{T}} := \bigsqcap_{C_i \dot{\sqsubseteq} D_i \in \mathcal{T}} \neg C_i \sqcup D_i.$$

Let U be a role that does not occur in $\mathcal{T}$, C, D, or $\mathcal{R}$. We set

$$\mathcal{R}_U := \mathcal{R} \cup \{UU \dot{\sqsubseteq} U\} \cup \{R \dot{\sqsubseteq} U, \mathsf{Inv}(R) \dot{\sqsubseteq} U \mid R \textit{ occurs in } \mathcal{T}, C, D, \textit{ or } \mathcal{R}\}.$$

- *C is satisfiable w.r.t. $\mathcal{T}$ and $\mathcal{R}$ iff $C \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is satisfiable w.r.t. $\mathcal{R}_U$.*
- *D subsumes C with respect to $\mathcal{T}$ and $\mathcal{R}$ iff $C \sqcap \neg D \sqcap C_{\mathcal{T}} \sqcap \forall U.C_{\mathcal{T}}$ is unsatisfiable w.r.t. $\mathcal{R}_U$.*

The proof of Lemma 3 is similar to the ones that can be found in [67, 1]. Most importantly, it must be shown that, (a) if a $\mathcal{SH}^+\mathcal{IQ}$-concept C is satisfiable with respect to a terminology $\mathcal{T}$ and a generalised role hierarchy $\mathcal{R}$, then $C, \mathcal{T}$ have a *connected* model, i. e., a model where any two elements are connect by a role path over those roles occurring in C and $\mathcal{T}$, and (b) if y is reachable from x via a role path (possibly involving inverse roles), then $\langle x,y\rangle \in U^I$. These are easy consequences of the semantics and the definition of U.

Theorem 4 *Satisfiability and subsumption of $\mathcal{SH}^+\mathcal{IQ}$-concepts w.r.t. terminologies and generalised role hierarchies are polynomially reducible to (un)satisfiability of $\mathcal{SH}^+\mathcal{IQ}$-concepts w.r.t. generalised role hierarchies.*

3.2.1 Relationship with Grammar Logics

It is well-known that description and modal logics are closely related: for example, $\mathcal{ALC}$ can be viewed as a notational variant of the multi modal logic **K** [67, 17]. Related to the logics investigated here are *grammar logics* [21], a class of propositional multi modal logics where the accessibility relations are "axiomatised" through a grammar. More precisely, for σ_i, τ_j modal parameters, the production rule $\sigma_1 \dots \sigma_m \rightarrow \tau_1 \dots \tau_n$ can be viewed as an abbreviation for the axioms

$$[\sigma_1] \dots [\sigma_m] p \Rightarrow [\tau_1] \dots [\tau_n] p,$$

or as being a notational variant for the role inclusion axiom

$$\tau_1 \dots \tau_n \dot{\sqsubseteq} \sigma_1 \dots \sigma_m.$$

Analogously to the description logic case, the semantics of a grammar logic is defined by taking into account only those frames/relational structures that "satisfy the grammar".

Grammars are traditionally organised in (refinements of) the Chomsky hierarchy (see any textbook on formal languages, e.g., [28]), which also induces classes of grammar logics. For example, the class of *context free* grammar logics is the class of those propositional multi modal logics where the accessibility relations are axiomatised through a *context free* grammar. Unsurprisingly, the expressiveness of the grammars influences the expressiveness of the corresponding grammar logics. It was shown that satisfiability of *regular* grammar logics is ExpTime-complete [18], whereas this problem is undecidable for context free grammar logics [4, 5]. The latter result is closely related to the undecidability proof in [78]. In this paper, we are concerned with

- grammars that are *not* regular, but we do not allow for arbitrary context-free grammars (or any known normal forms thereof), and
- multi modal logics that provide a converse operator on modal parameters. That is, for σ a modal parameter, both $[\sigma]\varphi$ and $[\sigma^-]\varphi$ are formulae of our logic, and we allow mixtures of converse and atomic modal parameters in the rules of the grammar. Moreover, $\mathcal{SH}^+\mathcal{IQ}$ provides *graded* modalities that restrict the number of accessible worlds, see, e.g., [74, 44].

As a consequence of the first point, we could not re-use the technique from [4, 5] for our undecidability proof: we could not reduce the emptiness problem for the intersection of context-free grammars to the satisfiability of $\mathcal{SH}^+IQ$-concepts because $\mathcal{SH}^+IQ$'s syntactic restriction on role inclusion axioms means that we cannot capture all context-free grammars. However, we can capture "some" context-freeness: our undecidability proof in Section 3.3 is by a reduction of the undecidable domino problem [8], and is heavily based on the language $\{(ab)^n(cd)^n \mid n \geq 0\}$ to enforce a model with a "grid" structure. Although we were not able to construct a grammar for this language directly using only productions of the form $R \to RS$ or $R \to SR$, we used a grammar G such that the language generated by G, when intersected with $(ab)^*(cd)^*$, equals $\{(ab)^n(cd)^n \mid n \geq 0\}$. This grammar G contains the four production rules

$$\begin{array}{lcl} D & \to & AD, \\ A & \to & AC, \\ C & \to & BC, \\ B & \to & BD, \ A \to a, \ldots D \to d \end{array}$$

and can be found in four versions as the last axioms of $\mathcal{R}_\mathcal{D}$ in Figure 2, where we use x_i, y_i, and their inverses instead of $A, \ldots, B$.

3.2.2 Role value maps

The role inclusion axioms we investigate here are closely related to *role value maps* [14, 69], i.e., concepts of the form $R_1 \ldots R_m \tilde{\sqsubseteq} S_1 \ldots S_n$ for R_i, S_i roles. The semantics of these concepts is defined as follows:

$$(R_1 \ldots R_m \tilde{\sqsubseteq} S_1 \ldots S_n)^I = \{x \in \Delta^I \mid (R_1 \ldots R_m)^I(x) \subseteq (S_1 \ldots S_n)^I(x)\},$$

where $(R_1 \ldots R_m)^I(x)$ denotes the set of those $y \in \Delta^I$ that are reachable from x via $R_1^I \circ \ldots \circ R_m^I$.

Thus the role inclusion axiom $RS \dot{\sqsubseteq} T$ is equivalent to the general concept inclusion axiom $\top \dot{\sqsubseteq} (RS \tilde{\sqsubseteq} T)$, i.e., both axioms have the same models. The role value maps used to show the undecidability of KL-ONE [69] are of a more general form than $(RS \tilde{\sqsubseteq} T)$, i.e., they use role chains of unbounded length on both sides of $\tilde{\sqsubseteq}$, and there is no direct translation of the undecidability proof in [69] to our logic.

3.3 $\mathcal{SH}^+IQ$ is undecidable

Due to the syntactic restriction on role inclusion axioms, neither the undecidability proof for $\mathcal{ALC}$ with context-free or linear grammars in [4, 5, 18] nor the one for $\mathcal{ALC}$ with role boxes [78] can be adapted to prove undecidability of $\mathcal{SH}^+IQ$ satisfiability. In the following, we reduce the (undecidable) domino problem [8] to $\mathcal{SH}^+IQ$ satisfiability. This problem asks whether, for a set of domino types, there exists a *tiling* of an $\mathbb{N}^2$ grid such that each point of the grid is covered with exactly one of the domino types, and adjacent dominoes are "compatible" with respect to some predefined criteria.

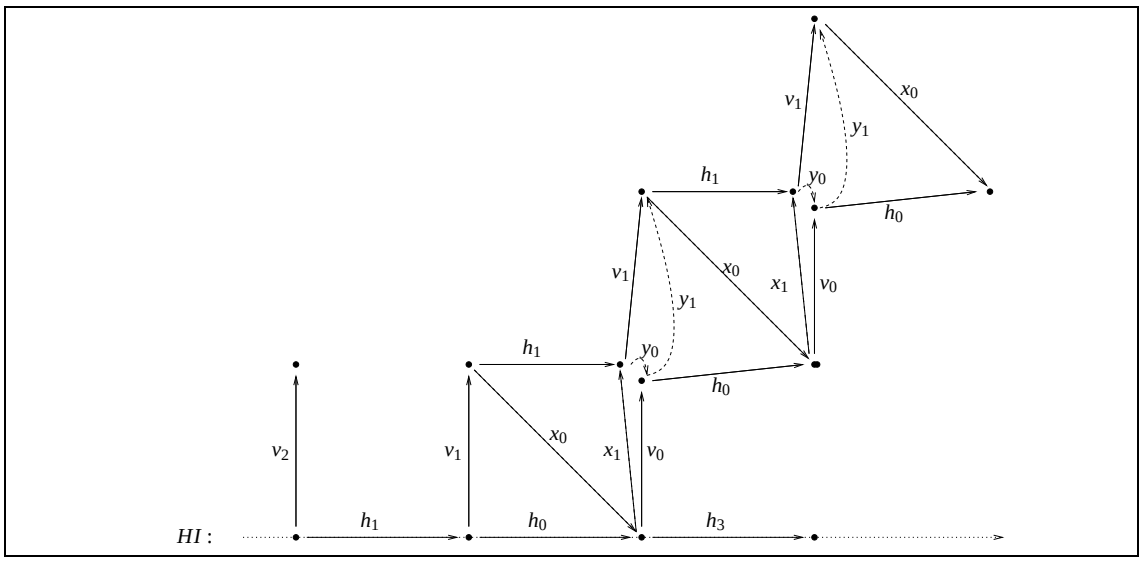

Figure 1: A staircase model and the implications of the last group of axioms in $\mathcal{R}_\mathcal{D}$.

Definition 5 *A domino system $\mathcal{D} = (D, H, V)$ consists of a non-empty set of domino types $D = \{D_1, \ldots, D_n\}$, and of 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 $\mathcal{D}$, the problem of determining if there exists a tiling for $\mathcal{D}$ is known to be undecidable [8].

In Figure 2, for a domino system $\mathcal{D}$, we define a $\mathcal{SH}^+ IQ$-concept $C_\mathcal{D}$, a terminology $\mathcal{T}_D$ (that can be internalised, see Theorem 4), and a generalised role hierarchy $\mathcal{R}_D$ such that $\mathcal{D}$ has a tiling iff $C_\mathcal{D}$ is satisfiable w.r.t. $\mathcal{R}_\mathcal{D}$ and $\mathcal{T}_\mathcal{D}$. For better readability, we use $C \Rightarrow D$ as an abbreviation for $\neg C \sqcup D$.

Ensuring that a point is associated with exactly one domino type, that it has at most one vertical and at most one horizontal successor, and that these successors satisfy the horizontal and vertical matching conditions induced by H and V is standard and is done in the first GCI of $\mathcal{T}_\mathcal{D}$.

The next step is rather special: we do not force a grid structure, but a structure with "staircases", which is illustrated in Figure 1. To this purpose, we introduce four sub-roles $v_0, \ldots, v_3$ of v and four sub-roles $h_0, \ldots, h_3$ of h (see first line of $\mathcal{R}_\mathcal{D}$), and ensure that we only have "staircases". For each $i \in \{0, \ldots, 3\}$, an i-staircase is an alternating chain of v_i and h_i edges, without any other v_j- or h_j-successors. We use concepts HI and VI for points on the x-axis and y-axis respectively. At each point on the x-axis, two staircases start that need not meet again, one i-staircase starting with v_i and one $i \ominus 1$-staircase starting with $h_{i\ominus 1}$ (we use $\oplus$ and $\ominus$ to denote addition and subtraction modulo four); points on the

y-axis exhibit a symmetrical behaviour. The second GCI in $\mathcal{T}_\mathcal{D}$ introduces the concept I for all "initial" points, and then the third GCI ensures the staircase structure. It contains four implications: one for the vertical and one for the horizontal successorships, and these two implications once for the "non-initial" points (i.e., instances of $\neg I$), and once for the "initial points" (i.e., instances of HI or VI).

It remains to make sure that two elements b, b' representing the same point in the grid have the same domino type associated with them, where b and b' "represent the same point" if there is an n and an instance a of I such that each of them is reachable following a staircase starting at a for n steps, i.e., if there is

- a $v_i h_i$-path (resp. $h_i v_i$-path) of length $2n$ from a to b, and
- a $h_{i\ominus 1} v_{i\ominus 1}$-path (resp. $v_{i\oplus 1} h_{i\oplus 1}$-path) of length $2n$ from a to b'.

To this purpose, we add super roles x_i of h_i and y_i of v_i (for which we use dashed arrows in Figure 1), and the last group of role inclusion axioms in $\mathcal{R}_\mathcal{D}$. These role inclusion axioms ensure appropriate, additional role successorships between elements, and we use the additional roles x_i and y_i since we only want to have at most one v_i or h_i-successor. For each 2 staircases starting at the same element on one of the axes, these role inclusions ensure that each pair of elements representing the same point is related by y_i. That is, each element on an $i \oplus 1$-staircase that is an $x_{i\oplus 1}$-successor is related via y_i to the element on the i-staircase (which is a v_i-successor) representing the same point (see Figure 1).

To see this, start by considering the consequences of the role inclusion axioms for elements representing the four points $(1,0)$, $(2,0)$, $(1,1)$ and $(2,1)$. The elements representing $(1,0)$ and $(2,1)$ are related via $h_3 v_3$ and $v_0 h_0$, and as we cannot force these two paths to end in the same element, we might have two elements representing $(2,1)$. From the axioms $h_3 \dot{\sqsubseteq} x_3$, $v_3 \dot{\sqsubseteq} y_3$, $v_0 \dot{\sqsubseteq} y_0$ and $h_0 \dot{\sqsubseteq} x_0$, we see that $(1,0)$ and $(2,1)$ are also related via $x_3 y_3$ and $y_0 x_0$. Using the axiom $y_0^- x_3 \dot{\sqsubseteq} x_3$ first, then $x_0^- x_3 \dot{\sqsubseteq} x_0^-$, and finally $x_0^- y_3 \dot{\sqsubseteq} y_3$, we also see that, if there are two elements representing the point $(2,1)$, then they are related via y_3. Next, consider elements representing the four points $(2,1)$, $(2,2)$, $(3,1)$ and $(3,2)$, start with the axiom $y_0^- y_3 \dot{\sqsubseteq} y_0^-$, and then continue to work through the same role inclusion axioms as above. Repeating this argumentation, all elements on these two staircases that represent the same point can be seen to be related via the relation y_3. From an analogous argumentation for other pairs of staircases, using corresponding sets of role inclusion axioms, it follows that the last GCI in $\mathcal{T}_\mathcal{D}$ ensures that two elements representing the same point in the grid do indeed have the same domino type associated with them.

The above observations imply that the concept $C_\mathcal{D}$ is satisfiable w.r.t. $\mathcal{T}_\mathcal{D}$ and $\mathcal{R}_\mathcal{D}$ iff $\mathcal{D}$ has a solution. Hence, together with Theorem 4, we have the following:

Theorem 6 *Satisfiability of $\mathcal{SH}^+IQ$-concepts w.r.t. generalized role hierarchies is undecidable.*

As mentioned above, the usage of inverse roles on the right hand side in RIAs of $\mathcal{R}_\mathcal{D}$ is of no importance: we can replace these RIAs with equivalent ones with role names on their right hand side, e.g., we can replace $x_{i\oplus 1}^- x_i \dot{\sqsubseteq} x_{i\oplus 1}^-$ with $x_i^- x_{i\oplus 1} \dot{\sqsubseteq} x_{i\oplus 1}$. However, we have chosen the representation in Figure 2 to make the relationship with the grammar from Section 3.2.1 more clear.

$$
\begin{array}{lll}
\mathcal{T}_{\mathcal{D}} & := \{ & \top \doteq (\bigsqcup_{1\le i\le n} D_i) \sqcap (\bigsqcap_{1\le i<j\le n} \neg(D_i \sqcap D_j)) \sqcap \\
& & \quad \bigsqcap_{1\le i\le n} D_i \Rightarrow ((\leqslant 1 v.\top) \sqcap (\forall v. \bigsqcup_{(D_i,D_j)\in V} D_j)) \sqcap \\
& & \quad \bigsqcap_{1\le i\le n} D_i \Rightarrow ((\leqslant 1 h.\top) \sqcap (\forall h. \bigsqcup_{(D_i,D_j)\in H} D_j), \\
& & I \doteq HI \sqcup VI, \\
& & \top \doteq \bigsqcap_{0\le i\le 3} (\exists v_i^-.\top \sqcap \neg I) \Rightarrow \big(\exists h_i.\neg I \sqcap \bigsqcap_j \forall v_j.\bot \sqcap \bigsqcap_{j\neq i} \forall h_j.\bot\big) \sqcap \\
& & \qquad (\exists h_i^-.\top \sqcap \neg I) \Rightarrow \big(\exists v_i.\neg I \sqcap \bigsqcap_{j\neq i} \forall v_j.\bot \sqcap \bigsqcap_j \forall h_j.\bot\big) \sqcap \\
& & \qquad (\exists h_i^-.\top \sqcap HI) \Rightarrow \big(\exists v_i.\neg I \sqcap \exists h_{i\ominus 1}.HI \sqcap \\
& & \qquad\qquad \bigsqcap_{j\neq i\ominus 1} \forall h_j.\bot \sqcap \bigsqcap_{j\neq i} \forall v_j.\bot\big) \sqcap \\
& & \qquad (\exists v_i^-.\top \sqcap VI) \Rightarrow \big(\exists h_i.\neg I \sqcap \exists v_{i\oplus 1}.VI \sqcap \\
& & \qquad\qquad \bigsqcap_{j\neq i\oplus 1} \forall v_j.\bot \sqcap \bigsqcap_{j\neq i} \forall h_j.\bot\big), \\
& & \top \doteq \bigsqcap_{0\le i\le 3} \bigsqcap_{1\le j\le n} \exists x_{i\oplus 1}^-.\top \Rightarrow (D_j \Rightarrow \forall y_i.D_j) \} \\
C_{\mathcal{D}} & := & HI \sqcap VI \sqcap \exists h_0.HI \sqcap \exists v_1.VI \\
\mathcal{R}_{\mathcal{D}} & := & \{v_i \dot{\sqsubseteq} v,\ h_i \dot{\sqsubseteq} h,\ v_i \dot{\sqsubseteq} y_i,\ h_i \dot{\sqsubseteq} x_i, \mid 0 \le i \le 3\} \cup \\
& & \{\ x_{i\oplus 1}^- y_i \ \dot{\sqsubseteq}\ y_i, \\
& & \ \ x_{i\oplus 1}^- x_i \ \dot{\sqsubseteq}\ x_{i\oplus 1}^-, \\
& & \ \ y_{i\oplus 1}^- x_i \ \dot{\sqsubseteq}\ x_i, \\
& & \ \ y_{i\oplus 1}^- y_i \ \dot{\sqsubseteq}\ y_{i\oplus 1}^- \mid 0 \le i \le 3\}
\end{array}
$$

Figure 2: Reduction terminology, generalised role hierarchy, and concept.

3.4 $\mathcal{RIQ}$ is decidable

In this section, we show that $\mathcal{SHIQ}$ with *regular* role hierarchies is decidable, where "regular" is both a restriction and a generalisation of "generalised". On the one hand, we restrict role hierarchies to be *acyclic*, where acyclic role hierarchies still allow for RIAs of the form $RS \dot{\sqsubseteq} S$, $SR \dot{\sqsubseteq} S$, $SS \dot{\sqsubseteq} S$, and $R^- \dot{\sqsubseteq} R$. Moreover, for convenience of proofs, we restrict our attention to RIAs with a role *name* on their right hand side. As mentioned above, this is of no importance. On the other hand, we also allow for axioms of the form $R_1 \dots R_n S \dot{\sqsubseteq} S$ and $SR_1 \dots R_n \dot{\sqsubseteq} S$ (for $\mathcal{SH}^+\mathcal{IQ}$, we restricted n to be 1). Finally, we also allow for statements that force roles to be *symmetric*, i.e., in contrast to the decidable case in [33], regularity also allows for RIAs of the form $\mathsf{Inv}(S) \dot{\sqsubseteq} S$.

We present a tableau-based algorithm that decides satisfiability of $\mathcal{RIQ}$-concepts w.r.t. regular role hierarchies, and therefore also subsumption in $\mathcal{RIQ}$ and, with The-

orem 4, both inferences w.r.t. terminologies. The FaCT system [30] was extended to use the algorithm presented in this section, and the empirical results are reported in Section 3.5.

The algorithm tries to construct, for a $\mathcal{RIQ}$-concept C, a *tableau* for C, that is, an abstraction of a model of C. Given the appropriate notion of a tableau, it is then quite straightforward to prove that the algorithm is a decision procedure for $\mathcal{RIQ}$-satisfiability. Before specifying this algorithm, we translate the role hierarchy into non-deterministic automata which are used both in the definition of a tableau and in the tableau algorithm. Intuitively, an automaton is used to memorise the path between an object x that has to satisfy a concept of the form $\forall R.C$ and other objects, and then to determine which of these objects must satisfy C.[14]

In the following definition of general role hierarchies, we use a strict partial order $\prec$ (irreflexive, transitive, and antisymmetric) on roles to ensure acyclicity.

Definition 7 *Let $\prec$ be a strict partial order on role names. A RIA $w \dot{\sqsubseteq} R$ is* $\prec$-regular *if*

- *R is a role name,*
- *$w = RR$,*
- *$w = R^-$,*
- *$w = S_1 \dots S_n$ and $S_i \prec R$, for all $1 \leq i \leq n$,*
- *$w = RS_1 \dots S_n$ and $S_i \prec R$, for all $1 \leq i \leq n$, or*
- *$w = S_1 \dots S_n R$ and $S_i \prec R$, for all $1 \leq i \leq n$.*

A role hierarchy $\mathcal{R}$ is regular *if there exists a strict partial order $\prec$ such that each RIA in $\mathcal{R}$ is $\prec$-*regular. *The semantics is defined analogously to the semantics of generalised role hierarchies, i.e., I satisfies a RIA $w \dot{\sqsubseteq} R$ if $w^I \subseteq R^I$.*

$\mathcal{RIQ}$ is obtained from $\mathcal{SH}^+\mathcal{IQ}$ by replacing generalised role hierarchies with regular role hierarchies, where simple role names *are inductively defined as follows:*[15]

- *every role name that does not occur on the right hand side of a RIA is simple,*
- *a role name S is simple if, for each $w \dot{\sqsubseteq} S \in \mathcal{R}$, $w = R$ for R a simple role or the inverse of a simple role.*

An inverse role S^- is simple if S is simple.

Please note that, due to the third restriction in the definition of R-compatibility, we also restrict $\dot{\sqsubseteq}^*$ to be acyclic. However, this is not a serious restriction since, for $\mathcal{R}$ containing $\dot{\sqsubseteq}^*$ cycles, we can simply choose one role R from each cycle and replace all other roles on this cycle with R, both in the input role hierarchy and the input concept.

For the following considerations, it is worthwhile to recall that, for $w = R_1 \dots R_m$ and R_i roles, $\mathsf{Inv}(w) = \mathsf{Inv}(R_m) \dots \mathsf{Inv}(R_1)$. The following Lemma is a direct consequence of the definition of the semantics.

[14] This technique together with the relationship between automata and regular languages is the reason why we called these role hierarchies "regular".

[15] We need to re-define "simple" roles because of the more general form of RIAs.

Lemma 8 *If I is a model of $\mathcal{R}$ with $S^- \dot{\sqsubseteq} S \in \mathcal{R}$ and $w \dot{\sqsubseteq} S \in \mathcal{R}$, then $\mathsf{Inv}(w)^I \subseteq S^I$.*

3.4.1 Translating RIAs into automata

Next, we will define, for a regular role hierarchy $\mathcal{R}$ and a (possibly inverse) role S occurring in $\mathcal{R}$, a non-deterministic finite automaton (NFA) $\mathcal{B}_S$ which captures all implications between (paths of) roles and S that are consequences of $\mathcal{R}$. To make this clear, before we define $\mathcal{B}_S$, we formulate the lemma which we are going to prove for it.

Proposition 9 *I is a model of $\mathcal{R}$ if and only if, for each (possibly inverse) role S occurring in $\mathcal{R}$, each word $w \in L(\mathcal{B}_S)$, and each $\langle x,y\rangle \in w^I$, we have $\langle x,y\rangle \in S^I$.*

In [33], to construct a similar automaton for a more restricted logic, we first unfolded $\mathcal{R}$ into a set of implications between regular expressions, and then constructed the automata from these implications. Here, we show how to build these automata directly, which yields an easier construction.

In the following, we use NFAs with ε-transitions in a rather informal way (see, e.g., [28] for a more details), e.g., we use $p \xrightarrow{R} q$ to denote that there is a transition from a state p to a state q with the letter R instead of introducing transition relations formally. The automata $\mathcal{B}_S$ are defined in three steps.

Definition 10 *Let C_0 be a $\mathcal{RIQ}$-concept and $\mathcal{R}$ a regular role hierarchy.*

For each role name R occurring in $\mathcal{R}$ or C_0, we first define the NFA $\mathcal{A}_R$ as follows: $\mathcal{A}_R$ contains a state i_R and a state f_R with the transition $i_R \xrightarrow{R} f_R$. The state i_R is the only initial state and f_R is the only final state. Moreover, for each $w \dot{\sqsubseteq} R \in \mathcal{R}$, $\mathcal{A}_R$ contains the following states and transitions:

1. *if $w = RR$, then $\mathcal{A}_R$ contains $f_R \xrightarrow{\varepsilon} i_R$, and*
2. *if $w = R_1 \cdots R_n$ and $R_1 \neq R \neq R_n$, then $\mathcal{A}_R$ contains*

$$i_R \xrightarrow{\varepsilon} i_w \xrightarrow{R_1} f_w^1 \xrightarrow{R_2} f_w^2 \xrightarrow{R_3} \ldots \xrightarrow{R_n} f_w^n \xrightarrow{\varepsilon} f_R,$$

3. *if $w = RR_2 \cdots R_n$, then $\mathcal{A}_R$ contains*

$$f_R \xrightarrow{\varepsilon} i_w \xrightarrow{R_2} f_w^2 \xrightarrow{R_3} f_w^3 \xrightarrow{R_4} \ldots \xrightarrow{R_n} f_w^n \xrightarrow{\varepsilon} f_R,$$

4. *if $w = R_1 \cdots R_{n-1}R$, then $\mathcal{A}_R$ contains*

$$i_R \xrightarrow{\varepsilon} i_w \xrightarrow{R_1} f_w^1 \xrightarrow{R_2} f_w^2 \xrightarrow{R_3} \ldots \xrightarrow{R_{n-1}} f_w^{n-1} \xrightarrow{\varepsilon} i_R,$$

where all f_w^i, i_w are assumed to be distinct.

In the next step, we use a mirrored copy *of NFAs: this is a copy of an NFA in which we have carried out the following modifications: we*

- *make final states to non-final but initial states,*

- *make initial states to non-initial but final states,*
- *replace each transition* $p \xrightarrow{S} q$ *for S a (possibly inverse) role S with* $q \xrightarrow{\mathsf{Inv}(S)} p$*, and*
- *replace each transition* $p \xrightarrow{\varepsilon} q$ *with* $q \xrightarrow{\varepsilon} p$*.*

Secondly, we define the NFAs $\hat{\mathcal{A}}_R$ *as follows:*

- *if* $R^- \dot{\sqsubseteq} R \notin \mathcal{R}$*, then* $\hat{\mathcal{A}}_R := \mathcal{A}_R$*,*
- *if* $R^- \dot{\sqsubseteq} R \in \mathcal{R}$*, then* $\hat{\mathcal{A}}_R$ *is obtained as follows: first, take the disjoint union*[16] *of* $\mathcal{A}_S$ *with a mirrored copy of* $\mathcal{A}_S$*. Secondly, make* i_R *the only initial state,* f_R *the only final state. Finally, for* f'_R *the copy of* f_R *and* i'_R *the copy of* i_R*, add transitions* $i_R \xrightarrow{\varepsilon} f'_R$*,* $f'_R \xrightarrow{\varepsilon} i_R$*,* $i'_R \xrightarrow{\varepsilon} f_R$*, and* $f_R \xrightarrow{\varepsilon} i'_R$*.*

Thirdly, the NFAs $\mathcal{B}_R$ *are defined inductively over* $\prec$*:*

- *if R is minimal w.r.t.* $\prec$ *(i.e., there is no* R' *with* $R' \prec R$*), we set* $\mathcal{B}_R := \hat{\mathcal{A}}_R$*.*
- *otherwise,* $\mathcal{B}_R$ *is the disjoint union of* $\hat{A}_R$ *with a copy* $\mathcal{B}'_S$ *of* $\mathcal{B}_S$ *for each transition* $p \xrightarrow{S} q$ *in* $\hat{A}_R$ *with* $S \neq R$*. Moreover, for each such transition, we add* ε*-transitions from p to the initial state in* $\mathcal{B}'_S$ *and from the final state in* $\mathcal{B}'_S$ *to q, and we make* i_R *the only initial state and* f_R *the only final state in* $\mathcal{B}_R$*.*

Finally, the automaton $\mathcal{B}_{R^-}$ *is a mirrored copy of* $\mathcal{B}_R$*.*

Please note that the inductive definition $\mathcal{B}_R$ is well-defined since the acyclic relation $\prec$ is used to restrict the dependencies between roles.

We have kept the construction of $\mathcal{B}_S$ as simple as possible. If one wants to construct an equivalent NFA without ε-transitions or which is deterministic, then there are well-known techniques to do this [28]. Recall that elimination of ε-transitions can be carried out without increasing the number of an automaton's states, whereas determinisation might yield an exponential blow-up.

Lemma 11 *For R a role, the size of* $\mathcal{B}_R$ *is bounded exponentially in the depth*

$$d_{\mathcal{R}} := \max\{n \mid \text{there are } S_1 \prec \ldots \prec S_n, u_i, v_i \text{ with } u_i S_{i-1} v_i \dot{\sqsubseteq} S_i \in \mathcal{R}\}$$

and thus in the size of $\mathcal{R}$*. Moreover, there are* $\mathcal{R}$ *and R such that the number of states in* $\mathcal{B}_R$ *is* $2^{d_{\mathcal{R}}}$*.*

Proof: Obviously, the size of $\mathcal{A}_R$ and $\hat{\mathcal{A}}_R$ is linear in

$$b_{\mathcal{R}} = \max\{|w_1| + \ldots + |w_k| \mid \text{there is } S \text{ with } w_i \dot{\sqsubseteq} S \in \mathcal{R} \text{ for all } 1 \leq i \leq n\}.$$

Each automaton $\mathcal{B}_R$ is a "tree" of automata $\mathcal{A}_S$ whose

- outdegree is bounded by $b_{\mathcal{R}}$ and

[16] A disjoint union of two automata is the disjoint union of their states, transition relations, etc.

- whose depth is bounded by $d_{\mathcal{R}}$.

Hence the number of $\mathcal{B}_R$'s states is bounded exponentially in $d_{\mathcal{R}}$ and, since $d_{\mathcal{R}}$ is linear in the size of $\mathcal{R}$, also bounded exponentially in the size of $\mathcal{R}$.

Next, it is easily verified that, for the following regular role hierarchy $\mathcal{R}_n$, the automaton $\mathcal{B}_{S_n}$ has 2^{n+1} states and the size of $\mathcal{R}_n$ is linear in n:

$$\mathcal{R}_n = \{S_{i-1}S_i \dot{\sqsubseteq} S_i, \quad S_iS_{i-1} \dot{\sqsubseteq} S_i \mid 1 \leq i \leq n\}$$

We will consider ways to avoid this exponential blow-up in Section 3.4.4, and continue with the proof of Proposition 9. In this proof, we will use the following lemma, which is an immediate consequence of the definition of $\mathcal{B}_S$ and of mirrored copies of $\mathcal{B}_S$.

Lemma 12 *1. $S \in L(\mathcal{B}_S)$ and, if $w \dot{\sqsubseteq} S \in R$, then $w \in L(\mathcal{B}_S)$.*

2. If S is a simple role, then $L(\mathcal{B}_S) = \{R \mid R \sqsubseteq^{*} S\}$.

3. If $\overleftarrow{\mathcal{A}}$ is a mirrored copy of an NFA $\mathcal{A}$, then $L(\overleftarrow{\mathcal{A}}) = \{\mathsf{Inv}(w) \mid w \in L(\mathcal{A})\}$.

Proof of Proposition 9. The "if" direction is easily proved by contraposition. If I is *not* a model of $\mathcal{R}$, then there is some RIA $w \dot{\sqsubseteq} S \in \mathcal{R}$ not satisfied by I. Hence there are some x,y such that $\langle x,y\rangle \in w^I$ but $\langle x,y\rangle \notin S^I$. By Lemma 12.1, $w \in L(\mathcal{B}_S)$, and we are done.

For the "only-if" direction, let I be a model of $\mathcal{R}$, S a role, $w \in L(\mathcal{B}_S)$, and $\langle x,y\rangle \in w^I$. We prove $\langle x,y\rangle \in S^I$ by well-founded induction on $\prec$. Obviously, we can restrict our attention to a role name S due to Lemma 12.3 and since $\mathcal{B}_{S^-}$ is defined as a mirrored copy of $\mathcal{B}_S$.

First, we observe that $w \in L(\mathcal{B}_S)$ induces a decomposition $w = w_1 \dots w_k$ and word $\hat{w} = S_1 \dots S_k$ such that

- $S_i \prec S$ or $S_i = S$ for all $1 \leq i \leq k$,
- $\hat{w} \in L(\hat{A}_S)$, and
- $w_i \in L(\mathcal{B}_{S_i})$.

Next, $\langle x,y\rangle \in w^I$ implies that there are x_i with $x = x_0$, $y = x_k$, and $\langle x_i, x_{i+1}\rangle \in w^I_{i+1}$, for each $0 \leq i < k$. By induction, $\langle x_i, y_i\rangle \in S_i^I$ and thus $\langle x,y\rangle \in \hat{w}^I$.

1. If $SS \dot{\sqsubseteq} S \notin \mathcal{R}$ and $S^- \dot{\sqsubseteq} S \notin \mathcal{R}$, then, by construction, $\hat{w}$ is of the form

$$\hat{w} = u_1 \dots u_m x v_1 \dots v_n \text{ and } \begin{array}{rl} u_i S \dot{\sqsubseteq} S \in \mathcal{R}, & \text{for each } 1 \leq i \leq m \\ x \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } x = S & \\ S v_j \dot{\sqsubseteq} S \in \mathcal{R}, & \text{for each } 1 \leq j \leq n \end{array}$$

Thus I being a model of $\mathcal{R}$ implies that $\langle x,y\rangle \in S^I$.

2. If $SS \dot{\sqsubseteq} S \in \mathcal{R}$ and $S^- \dot{\sqsubseteq} S \notin \mathcal{R}$, then, by construction, $\hat{w}$ is of the form

$$\begin{aligned}
\hat{w} = (u_1^{(1)} \dots u_{m_1}^{(1)} x^{(1)} v_1^{(1)} \dots v_{n_1}^{(1)}) \dots (u_1^{(\ell)} \dots u_{m_\ell}^{(\ell)} x^{(\ell)} v_1^{(\ell)} \dots v_{n_\ell}^{(\ell)}) \text{ and} \\
u_i^{(k)} S \dot{\sqsubseteq} S \in \mathcal{R}, \quad \text{for each } 1 \le i \le m,\ 1 \le k \le \ell \\
x^{(k)} \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } x^{(k)} = S \quad \text{for each } 1 \le k \le \ell \\
S v_j^{(k)} \dot{\sqsubseteq} S \in \mathcal{R}, \quad \text{for each } 1 \le j \le n,\ 1 \le k \le \ell
\end{aligned}$$

   Again, I being a model of $\mathcal{R}$ implies that $\langle x,y \rangle \in S^I$.

3. If $SS \dot{\sqsubseteq} S \notin \mathcal{R}$ and $S^- \dot{\sqsubseteq} S \in \mathcal{R}$, then $\mathcal{B}_S$ is the disjoint union of $\mathcal{A}_S$ with a mirrored copy of $\mathcal{A}_S$ and additional ε-transitions between the final and initial state and their copies. By construction, we have

$$\begin{aligned}
\hat{w} = u_1 \dots u_m x v_1 \dots v_n \text{ and} \\
u_i S \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } S\,\mathsf{Inv}(u_i) \dot{\sqsubseteq} S \in R \quad \text{for each } 1 \le i \le m \\
x \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } \mathsf{Inv}(x) \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } x = S \text{ or } x = S^- \\
S v_j \dot{\sqsubseteq} S \in \mathcal{R} \text{ or } \mathsf{Inv}(v_j) S \dot{\sqsubseteq} S \in R, \quad \text{for each } 1 \le j \le n
\end{aligned}$$

   In both cases, I being a model of $\mathcal{R}$ implies that $\langle x,y \rangle \in S^I$.

4. If $SS \dot{\sqsubseteq} S \in \mathcal{R}$ and $S^- \dot{\sqsubseteq} S \in \mathcal{R}$, then we are in a mixture of the cases (2) and (3), i.e.,

$$\hat{w} = \hat{w}_1 \dots \hat{w}_r$$

   and each $\hat{w}_i$ is accepted by a run through $\mathcal{B}_S$ which neither uses the ε-transition from f_S to i_S nor the corresponding one in the mirrored copy of $\hat{A}_S$. We can decompose each $\hat{w}_i$ as we have decomposed $\hat{w}$ in Case (3), and conclude that I being a model of $\mathcal{R}$ implies that $\langle x,y \rangle \in S^I$. ∎

3.4.2 A Tableau for $\mathcal{R}IQ$

In the following, if not stated otherwise, C, D (possibly with subscripts) denote $\mathcal{R}IQ$-concepts, R, S (possibly with subscripts) roles, and $\mathcal{R}$ a regular role hierarchy.

We start by defining $\mathsf{fclos}(C_0, \mathcal{R})$, the *closure* of a concept C w.r.t. a regular role hierarchy $\mathcal{R}$. Intuitively, this contains all relevant sub-concepts of C together with universal value restrictions over sets of role paths described by an NFA. We use NFAs in universal value restrictions to memorise the path between an object that has to satisfy a value restriction and other objects. To do this, we "push" this NFA-value restriction along this path while the NFA gets "updated" with the path taken so far. For this "update", we use the following definition.

Definition 13 *For $\mathcal{B}$ an NFA and q a state of $\mathcal{B}$, $\mathcal{B}(q)$ denotes the NFA obtained from $\mathcal{B}$ by making q the (only) initial state of $\mathcal{B}$, and we use $q \xrightarrow{S} q' \in \mathcal{B}$ to denote that $\mathcal{B}$ has a transition $q \xrightarrow{S} q'$.*

Without loss of generality, we assume all concepts to be in NNF, that is, negation occurs in front of concept names only. Any $\mathcal{R}IQ$-concept can easily be transformed into an equivalent one in NNF by pushing negations inwards using a combination of DeMorgan's laws and the following equivalences:

$$\begin{array}{rcl@{\qquad}rcl} \neg(\exists R.C) & \equiv & (\forall R.\neg C) & \neg(\forall R.C) & \equiv & (\exists R.\neg C) \\ \neg(\leqslant nR.C) & \equiv & (\geqslant (n+1)R.C) & \neg(\geqslant (n+1)R.C) & \equiv & (\leqslant nR.C) \\ & & & \neg(\geqslant 0R.C) & \equiv & \bot \end{array}$$

We use $\dot{\neg}C$ for the NNF of $\neg C$. Obviously, the length of $\dot{\neg}C$ is linear in the length of C.

For a concept C_0, $\mathsf{clos}(C_0)$ is the smallest set that contains C_0 and that is closed under sub-concepts and $\dot{\neg}$. The set $\mathsf{fclos}(C_0,\mathcal{R})$ is then defined as follows:

$$\mathsf{fclos}(C_0,\mathcal{R}) := \mathsf{clos}(C_0) \cup \{\forall \mathcal{B}_S(q).D \mid \forall S.D \in \mathsf{clos}(C_0) \text{ and } \mathcal{B}_S \text{ has a state } q\}.$$

It is not hard to show and well-known that the size of $\mathsf{clos}(C_0)$ is linear in the size of C_0. For the size of $\mathsf{fclos}(C_0,\mathcal{R})$, we have seen in Lemma 11 that, for a role S, the size of $\mathcal{B}_S$ can be exponential in the depth of $\mathcal{R}$. Since there are at most linearly many concepts $\forall S.D$, this yields a bound for the cardinality of $\mathsf{fclos}(C_0,\mathcal{R})$ that is exponential in the depth of $\mathcal{R}$ and linear in the size of C_0. Investigating whether this exponential blow-up can be avoided will be part of future work. So far, we only define in Section 3.4.4 a further syntactic restriction which avoids this exponential blow-up.

We are now ready to define tableaux as a useful abstraction of models.

Definition 14 *$T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a* tableau *for C_0 w.r.t. $\mathcal{R}$ iff*

- *$\mathbf{S}$ is a non-empty set,*
- *$\mathcal{L} : \mathbf{S} \to 2^{\mathsf{fclos}(C_0,\mathcal{R})}$ maps each element in $\mathbf{S}$ to a set of concepts and*
- *$\mathcal{E} : \mathbf{R}_{C_0,\mathcal{R}} \to 2^{\mathbf{S}\times\mathbf{S}}$ maps each role to a set of pairs of elements in $\mathbf{S}$.*

Furthermore, for all $s,t \in \mathbf{S}$, $C, C_1, C_2 \in \mathsf{fclos}(C_0,\mathcal{R})$, and $R,S \in \mathbf{R}_{C_0,\mathcal{R}}$, T satisfies:

(P0) *there is some $s \in \mathbf{S}$ with $C_0 \in \mathcal{L}(s)$,*

(P1) *if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,*

(P2) *if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,*

(P3) *if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,*

(P4a) *if $\forall \mathcal{B}(p).C \in \mathcal{L}(s)$, $\langle s,t\rangle \in \mathcal{E}(S)$, and $p \xrightarrow{S} q \in \mathcal{B}(p)$, then $\forall \mathcal{B}(q).C \in \mathcal{L}(t)$,*

(P4b) *if $\forall \mathcal{B}.C \in \mathcal{L}(s)$ and $\varepsilon \in L(\mathcal{B})$, then $C \in \mathcal{L}(s)$,*

(P5) *if $\exists S.C \in \mathcal{L}(s)$, then there is some t with $\langle s,t\rangle \in \mathcal{E}(S)$ and $C \in \mathcal{L}(t)$,*

(P6) *if $\forall S.C \in \mathcal{L}(s)$, then $\forall \mathcal{B}_S.C \in \mathcal{L}(s)$,*

(P7) *$\langle x,y\rangle \in \mathcal{E}(R)$ iff $\langle y,x\rangle \in \mathcal{E}(\mathsf{Inv}(R))$,*

(P8) *if $(\leqslant nS.C) \in \mathcal{L}(s)$, then $\sharp S^T(s,C) \leqslant n$,*

(P9) *if* $(\geqslant nS.C) \in \mathcal{L}(s)$, *then* $\sharp S^T(s,C) \geqslant n$,

(P10) *if* $(\leqslant nS.C) \in \mathcal{L}(s)$ *and* $\langle s,t\rangle \in \mathcal{E}(S')$ *for some* $S' \in L(\mathcal{B}_S)$, *then* $C \in \mathcal{L}(t)$ *or* $\doteq\neg C \in \mathcal{L}(t)$,

where $S^T(s,C) := \{t \in \mathbf{S} \mid \langle s,t\rangle \in \mathcal{E}(S')$ *for some* $S' \in L(\mathcal{B}_S)$ *and* $C \in \mathcal{L}(t)\}$.

Lemma 15 *A $\mathcal{R}IQ$-concept C_0 is satisfiable w.r.t. $\mathcal{R}$ iff there exists a tableau for C_0 w.r.t. $\mathcal{R}$.*

Proof: For the *if* direction, let $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ be a tableau for C_0 w.r.t. $\mathcal{R}$. We extend the relational structure of T and then prove that this indeed gives a model. More precisely, a model $I = (\Delta^I, \cdot^I)$ of D and $\mathcal{R}$ can be defined as follows: we set $\Delta^I := \mathbf{S}$, $A^I := \{s \mid A \in \mathcal{L}(s)\}$ for concept names A in $\mathsf{clos}(C_0)$, and for roles names R, we set

$$R^I := \{\langle s_0, s_n\rangle \in (\Delta^I)^2 \mid \text{there are } s_1,\ldots,s_{n-1} \text{ with } \langle s_i, s_{i+1}\rangle \in \mathcal{E}(S_{i+1}) \text{ for } 0 \leq i \leq n-1 \text{ and } S_1 \cdots S_n \in L(\mathcal{B}_R)\}$$

The semantics of complex concepts is given through the definition of the $\mathcal{R}IQ$ semantics. Due to Lemma 12.3 and (P7), the semantics of inverse roles can either be given directly as for role names, or by setting $(R^-)^I = \{\langle y,x\rangle \mid \langle x,y\rangle \in R^I\}$.

First, we show that I is a model of $\mathcal{R}$ and C_0. Due to Proposition 9, it suffices to prove that, for each (possibly inverse) role S, each word $w \in L(\mathcal{B}_S)$, and each $\langle x,y\rangle \in w^I$, we have $\langle x,y\rangle \in S^I$. Let $w \in L(\mathcal{B}_S)$ and $\langle x,y\rangle \in w^I$. For $w = S_1 \ldots S_n$, this implies the existence of y_i such that $y_0 = x$, $y_n = y$, and $\langle y_{i-1}, y_i\rangle \in S_i^I$ for each $1 \leq i \leq n$. For each i, we define a word w_i as follows:

- if $\langle y_{i-1}, y_i\rangle \in \mathcal{E}(S_i)$, then set $w_i := S_i$.
- otherwise, there is some $v_i = T_1^{(i)} \ldots T_{n_i}^{(i)} \in L(\mathcal{B}_{S_i})$ and there are $y_j^{(i)}$ such that $y_{i-1} = y_0^{(i)}$, $y_i = y_{n_i}^{(i)}$, and $\langle y_{j-1}^{(i)}, y_j^{(i)}\rangle \in \mathcal{E}(T_j^{(i)})$ for each $1 \leq j \leq n_i$. In this case, we set $w_i := v_i$.

Let $\hat{w} := w_1 \ldots w_n$. By construction of $\mathcal{B}_S$ from $\hat{\mathcal{A}}_S$, $w \in L(\mathcal{B}_S)$ implies that $\hat{w} \in L(\mathcal{B}_S)$. For $\hat{w} = U_1 \ldots U_{n'}$, we can thus re-name the y_i and $y_j^{(i)}$ to z_i such that we have $z_0 = x$, $z_n = y$, and $\langle z_{i-1}, z_i\rangle \in \mathcal{E}(U_i)$. Hence, by definition of $\cdot^I$, we have $\langle x,y\rangle \in S^I$.

Secondly, we prove that I is a model of C_0. We show that $C \in \mathcal{L}(s)$ implies $s \in C^I$ for each $s \in \mathbf{S}$ and each $C \in \mathsf{clos}(C_0)$. Together with (P0), this implies that I is a model of C_0. This proof can be given by induction on the length of concepts, where we count neither negation nor integers in number restrictions. The only interesting cases are $C = (\leqslant nS.E)$ and $C = \forall S.E$ (for the other cases, see [42, 32]):

- If $(\leqslant nS.E) \in \mathcal{L}(s)$, then (P8) implies that $\#S^T(s,E) \leq n$. Moreover, since S is simple, Lemma 12.2 implies that $L(\mathcal{B}_S) = \{S' \mid S' \sqsubseteq^* S\}$, and thus (P10) implies that, for all t, if $\langle s,t\rangle \in S^I$, then $E \in \mathcal{L}(t)$ or $\doteq E \in \mathcal{L}(t)$. By induction $E^I = \{t \mid E \in \mathcal{L}(t)\}$, and thus $s \in (\leqslant nS.E)^I$.

- Let $\forall S.E \in \mathcal{L}(s)$ and $\langle s,t\rangle \in S^I$. From (P6) we have that $\forall \mathcal{B}_S.E \in \mathcal{L}(s)$. By definition of S^I, there are $S_1 \ldots S_n \in L(\mathcal{B}_S)$ and s_i with $s = s_0$, $t = s_n$, and $\langle s_{i-1}, s_i\rangle \in \mathcal{E}(S_i)$. Applying (P4a) n times, this yields $\forall \mathcal{B}_S(q).E \in \mathcal{L}(t)$ for q a final state of $\mathcal{B}_S$. Thus (P4b) implies that $E \in \mathcal{L}(t)$. By induction, $t \in E^I$, and thus $s \in (\forall S.E)^I$.

For the converse, for $I = (\Delta^I, \cdot^I)$ a model of C_0 w.r.t. $\mathcal{R}$, we define a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for C_0 and $\mathcal{R}$ as follows:

$$\begin{array}{rcl}
\mathbf{S} & := & \Delta^I, \\
\mathcal{E}(R) & := & R^I, \text{ and} \\
\mathcal{L}(s) & := & \{C \in \mathsf{clos}(C_0) \mid s \in C^I\} \cup \\
 & & \{\forall \mathcal{B}_S.C \mid \forall S.C \in \mathsf{clos}(C_0) \text{ and } s \in (\forall S.C)^I\} \cup \\
 & & \{\forall \mathcal{B}_R(q).C \in \mathsf{fclos}(C_0,\mathcal{R}) \mid \text{for all } S_1 \cdots S_n \in L(\mathcal{B}_R(q)), \\
 & & \qquad\qquad s \in (\forall S_1.\forall S_2.\cdots\forall S_n.C)^I \text{ and} \\
 & & \qquad\qquad \text{if } \varepsilon \in L(\mathcal{B}_R(q)), \text{ then } s \in C^I\}
\end{array}$$

We have to show that T satisfies each (Pi). We restrict our attention to the only new cases (P4) and (P6).

For (P6), if $\forall S.C \in \mathcal{L}(s)$, then $s \in (\forall S.C)^I$ and thus $\forall \mathcal{B}_S.C \in \mathcal{L}(s)$ by definition of T.

For (P4a), let $\forall \mathcal{B}(p).C \in \mathcal{L}(s)$ and $\langle s,t\rangle \in \mathcal{E}(S) = S^I$. Assume that there is a transition $p \xrightarrow{S} q$ in $\mathcal{B}(p)$ and $\forall \mathcal{B}(q).C \notin \mathcal{L}(t)$. By definition of T, this can have two reasons:

- there is a word $S_2 \ldots S_n \in L(\mathcal{B}(q))$ and $t \notin (\forall S_2 \ldots . \forall S_n.C)^I$. However, this implies that $SS_2 \ldots S_n \in L(\mathcal{B}(p))$ and thus that $s \in (\forall S.\forall S_2 \ldots . \forall S_n.C)^I$, which contradicts, together with $\langle s,t\rangle \in S^I$, the definition of the semantics of $\mathcal{R}\mathcal{I}\mathcal{Q}$ concepts.
- $\varepsilon \in L(\mathcal{B}(q))$ and $t \notin C^I$. This implies that $S \in L(\mathcal{B}(p))$ and thus contradicts $s \in (\forall S.C)^I$.

Hence $\forall \mathcal{B}(q).C \notin \mathcal{L}(t)$.

For (P4b), $\varepsilon \in L(\mathcal{B}(p))$ implies $s \in C^I$ by definition of T, and thus $C \in \mathcal{L}(s)$. ∎

3.4.3 The Tableau Algorithm

In this section, we present a tableau algorithm that tries to construct, for an input $\mathcal{R}\mathcal{I}\mathcal{Q}$-concept C_0 and a regular role hierarchy $\mathcal{R}$, a tableau for C_0 w.r.t. $\mathcal{R}$. We prove that this algorithm constructs a tableau for C_0 and $\mathcal{R}$ iff there exists a tableau for C_0 and $\mathcal{R}$, and thus decides satisfiability of $\mathcal{R}\mathcal{I}\mathcal{Q}$ concepts w.r.t. regular role hierarchies and, using Lemma 3, also w.r.t. terminologies.

This algorithm generates a *completion tree*, a structure that will be unravelled to an (infinite) tableau for the input concept. As usual, in the presence of transitive roles, *blocking* is employed to ensure termination of the algorithm. In the additional presence of inverse roles, blocking is *dynamic*, i.e., blocked nodes (and their sub-branches) can be un-blocked and blocked again later. In the further, additional presence of number restrictions, *pairs* of nodes are blocked rather than single nodes [42]. The blocking conditions as they are presented here are, clearly, too strict. As a consequence, blocking may occur

later than necessary, and thus we end up with a search space that is larger than necessary. In [32], we have shown how to loosen the blocking condition for $\mathcal{SHIQ}$ while retaining correctness of the algorithm. Here, we focus on the decidability of $\mathcal{RIQ}$, and defer a similar loosening for $\mathcal{RIQ}$ to future work.

Definition 16 *A* completion tree $\mathbf{T}$ *for a $\mathcal{RIQ}$ concept C_0 and a regular role hierarchy $\mathcal{R}$ is a tree, where each node x is labelled with a set $\mathcal{L}(x) \subseteq \mathsf{fclos}(C_0,\mathcal{R})$ and each edge $\langle x,y\rangle$ from a node x to its successor y is labelled with a non-empty set $\mathcal{L}(\langle x,y\rangle)$ of (possibly inverse) roles occurring in C_0 and $\mathcal{R}$. Finally, completion trees come with an explicit inequality relation $\neq$ on nodes which is implicitly assumed to be symmetric.*

*If $R \in \mathcal{L}(\langle x,y\rangle)$ for a node x and its successor y, then y is called an R-*successor *of x and x is called an $\mathsf{Inv}(R)$-*predecessor *of y. If y is an R-successor or an $\mathsf{Inv}(R)$-predecessor of x, then y is called an R-neighbour of x. Finally,* ancestor *is the transitive closure of* predecessor *and* descendant *is the transitive closure of* successor.

For a role S, a concept C and a node x in $\mathbf{T}$ we define $S^{\mathbf{T}}(x,C)$ by

$$S^{\mathbf{T}}(x,C) := \{y \mid \text{for some } S' \sqsubseteq^{*} S,\ y \text{ is an } S'\text{-neighbour of } x \text{ and } C \in \mathcal{L}(y)\}.$$

A node is blocked *iff it is either directly or indirectly blocked. A node x is* directly blocked *iff none of its ancestors are blocked, and it has ancestors x', y and y' such that*

1. *x is a successor of x' and y is a successor of y'* and
2. *$\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$* and
3. *$\mathcal{L}(\langle x',x\rangle) = \mathcal{L}(\langle y',y\rangle)$.*

If there are no descendants x'', y'' of x' and y' with these properties, then we say that y blocks *x.*

A node y is indirectly blocked *if one of its ancestors is blocked.*

For a node x, $\mathcal{L}(x)$ is said to contain a clash *if*

- *$\bot \in \mathcal{L}(x)$ or*
- *for some concept name A, $\{A, \neg A\} \subseteq \mathcal{L}(x)$ or*
- *there is some concept $(\leqslant nS.C) \in \mathcal{L}(x)$ and $\{y_0,\ldots,y_n\} \subseteq S^{\mathbf{T}}(x,C)$ with $y_i \neq y_j$ for all $0 \leq i < j \leq n$.*

A completion tree is clash-free *if none of its nodes contains a clash, and it is* complete *if no rule from Figure 3 can be applied to it.*

Given C_0 (in NNF) and $\mathcal{R}$, the algorithm initialises a completion tree consisting only of a root node x_0 labelled with $\{C_0\}$. Then this tree is expanded by repeatedly applying the expansion rules from Figure 3, stopping when a clash occurs. The algorithm answers "C_0 is satisfiable w.r.t. $\mathcal{R}$" iff the expansion rules can be applied in such a way that they yield a complete and clash-free completion tree, and "C_0 is unsatisfiable w.r.t. $\mathcal{R}$" otherwise.

All but the $\forall_i$-rules have been used before for fragments of $\mathcal{RIQ}$, e.g., $\mathcal{SHIQ}$ [34, 32], and the three $\forall_i$-rules are the obvious counterparts to the tableau conditions (P4a). (P4b), and (P6).

$\sqcap$-rule: if	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule: if	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and
	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{E\}$ for some $E \in \{C_1, C_2\}$
$\exists$-rule: if	$\exists S.C \in \mathcal{L}(x)$, x is not blocked, and
	x has no S-neighbour y with $C \in \mathcal{L}(y)$
then	create a new node y with
	$\mathcal{L}(\langle x, y \rangle) := \{S\}$ and $\mathcal{L}(y) := \{C\}$
$\forall_1$-rule: if	$\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and
	$\forall \mathcal{B}_S.C \notin \mathcal{L}(x)$
then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{\forall \mathcal{B}_S.C\}$
$\forall_2$-rule: if	$\forall \mathcal{B}(p).C \in \mathcal{L}(x)$, x is not indirectly blocked, $p \xrightarrow{S} q$ in $\mathcal{B}(p)$,
	and there is an S-neighbour y of x with $\forall \mathcal{B}(q).C \notin \mathcal{L}(y)$,
then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall \mathcal{B}(q).C\}$
$\forall_3$-rule: if	$\forall \mathcal{B}.C \in \mathcal{L}(x)$, x is not indirectly blocked, $\varepsilon \in L(\mathcal{B})$,
	and $C \notin \mathcal{L}(x)$
then	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$
X-rule: if	$(\leqslant n S.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and
	there is an S'-neighbour y of x with $S' \sqsubseteq\!\!\!\!^{*}\ S$
	and $\{C, \dot{\neg} C\} \cap \mathcal{L}(y) = \emptyset$
then	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \dot{\neg} C\}$
$\geqslant$-rule: if	$(\geqslant n S.C) \in \mathcal{L}(x)$, x is not blocked, and
	there are no $y_1, \ldots, y_n \in S^{\mathbf{T}}(x, C)$
	with $y_i \neq\!\!\!\!\cdot\ y_j$ for each $1 \leq i < j \leq n$
then	create n new nodes $y_1, \ldots, y_n$ with $\mathcal{L}(\langle x, y_i \rangle) = \{S\}$,
	$\mathcal{L}(y_i) = \{C\}$, and $y_i \neq\!\!\!\!\cdot\ y_j$ for $1 \leq i < j \leq n$.
$\leqslant$-rule: if	$(\leqslant n S.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and
	$\#S^{\mathbf{T}}(x, C) > n$, there are $y, z \in S^{\mathbf{T}}(x, C)$ with
	not $y \neq\!\!\!\!\cdot\ z$ and y is not an ancestor of z,
then	1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ and
	2. if z is an ancestor of x
	then $\mathcal{L}(\langle z, x \rangle) \longrightarrow \mathcal{L}(\langle z, x \rangle) \cup \mathsf{Inv}(\mathcal{L}(\langle x, y \rangle))$
	else $\mathcal{L}(\langle x, z \rangle) \longrightarrow \mathcal{L}(\langle x, z \rangle) \cup \mathcal{L}(\langle x, y \rangle)$
	3. remove y and the sub-tree below y

Figure 3: The Expansion Rules for the $\mathcal{RIQ}$ Tableau Algorithm.

As usual, we prove termination, soundness, and completeness of the tableau algorithm to show that it indeed decides satisfiability of $\mathcal{RIQ}$-concepts w.r.t. regular role hierarchies.

Lemma 17 *Let C_0 be a $\mathcal{RIQ}$-concept and $\mathcal{R}$ a regular role hierarchy. The tableau algorithm terminates when started for C_0 and $\mathcal{R}$.*

Proof: Let $m = \sharp\mathsf{fclos}(C_0, \mathcal{R})$, n the number of roles occurring in C_0 and $\mathcal{R}$, and $n_{\max} := \max\{n \mid (\geqslant nR.C) \in \mathsf{clos}(C_0)\}$. Termination is a consequence of the following properties of the expansion rules:

1. Nodes are labelled with subsets of $\mathsf{fclos}(C_0, \mathcal{R})$ and edges with sets of roles occurring in C_0 and $\mathcal{R}$, so there are at most 2^{2mn} different possible labellings for a pair of nodes and an edge. Therefore, if a path p is of length at least 2^{2mn}, the pair-wise blocking condition implies the existence of a node x on p such that x is blocked. Since a path on which nodes are blocked cannot become longer, paths are of length at most 2^{2mn}.

2. The expansion rules never remove labels from nodes in the tree, and the only rule that removes a node from the tree is the $\leqslant$-rule.

3. Only the $\exists$- or the $\geqslant$-rule generate new nodes, and each generation is triggered by a concept of the form $\exists R.C$ or $(\geqslant nR.C)$ in the label of a node x. Each of these concepts triggers at most once the generation of at most $n_{\max}$ R-successors y_i of x: note that if the $\leqslant$-rule subsequently causes an R-successor y_i of x to be removed, then x will have some R-neighbour z with $\mathcal{L}(z) \supseteq \mathcal{L}(y_i)$. This, together with the definition of a clash, implies that the rule application which led to the generation of y_i will not be repeated. Since $\mathsf{fclos}(C_0, \mathcal{R})$ contains a total of at most m $\exists R.C$, the out-degree of the tree is bounded by $mn_{\max}$. ∎

Lemma 18 *Let C_0 be a $\mathcal{RIQ}$-concept and $\mathcal{R}$ a regular role hierarchy. The expansion rules can be applied to C_0 and $\mathcal{R}$ such that they yield a complete and clash-free completion tree if and only if C_0 has a tableau w.r.t. $\mathcal{R}$.*

For the if direction, we can unravel a complete and clash-free completion tree **T** in a standard way into a tableau T, where the same technique as for $\mathcal{SHIQ}$ is used to make sure that (P9) is satisfied even if two "sibling" nodes are blocked by the same node. It is easily seen that the $\forall_i$ expansion rules make sure that the resulting structure indeed satisfies the new tableau condition (P4a), (P4b), and (P6).

For the only-if direction, we take a tableau I of C_0 and $\mathcal{R}$ and use it to steer the application of the non-deterministic rules, i.e., the $\sqcup$-, the X- and the $\leqslant$ -rule. To do this, while building the completion tree, we define a mapping π from the nodes of the completion tree into the tableau which satisfies the following three conditions:

$$\left.\begin{array}{l} \mathcal{L}(x) \subseteq \mathcal{L}(\pi(x)), \\ \text{if } y \text{ is an } S\text{-neighbour of } x, \text{ then } \langle \pi(x), \pi(y) \rangle \in \mathcal{E}(S), \text{ and} \\ x \neq\!\!\!\!\cdot\; y \text{ implies } \pi(x) \neq \pi(y). \end{array}\right\} \qquad (*)$$

We start with π mapping the root node to some tableau element s_0 with C_0 in its label, and prove that, if an expansion rule is applicable to **T**, then this rule can be applied in such a way that $(*)$ is preserved. As a consequence of this claim, (P1), (P8), and Lemma 17, we thus end with a complete and clash-free completion tree. For a full proof, see [31].

From Theorem 4, Lemma 15, 17, 18, and 18, we thus have the following theorem:

Theorem 19 *The tableau algorithm decides satisfiability and subsumption of $\mathcal{RIQ}$-concepts with respect to regular role hierarchies and terminologies.*

3.4.4 Avoiding the blow-up

In the previous section, we have presented an algorithm that decides satisfiability and subsumption of $\mathcal{RIQ}$-concepts with respect to regular role hierarchies and terminologies. Unfortunately, compared to similar algorithms that are implemented in state-of-the-art description logic reasoners [30, 57, 25] and behave well in many cases, we have here an exponential blow-up: the closure $\mathsf{fclos}(C_0, \mathcal{R})$ is exponential in the depth of $\mathcal{R}$ since we have "unfolded" the regular role hierarchy $\mathcal{R}$ into trees of NFAs. While investigating whether and how this exponential blow-up can be avoided, we observe that a further restriction of the syntax of regular role hierarchies avoids this blow-up:

A regular role hierarchy $\mathcal{R}$ is called *simple* when, for all S_i, T_i, n, m, $1 \leq i \leq n$, and $1 \leq j \leq m$, if

1. $u_i S_i v_i \dot{\sqsubseteq} S_{i+1} \in \mathcal{R}$ and $u'_j T_j v'_j \dot{\sqsubseteq} T_{j+1} \in \mathcal{R}$,
2. $S_i \neq S_{i+1}$ and $T_j \neq T_{j+1}$,
3. $S_n = T_m$ and $u_n \neq u'_m$,

then $S_i \neq T_j$.

For a *simple* regular role hierarchy $\mathcal{R}$, the size of each NFA $\mathcal{B}_R$ is only polynomial in the size of $\mathcal{R}$ since each NFA $\mathcal{B}_S$ occurs at most once in $\mathcal{B}_R$.

Lemma 20 *For a $\mathcal{RIQ}$-concept C_0 and a simple regular role hierarchy $\mathcal{R}$, the size of $\mathsf{fclos}(C_0, \mathcal{R})$ is polynomial in the size of C_0 and $\mathcal{R}$.*

Thus, for simple role hierarchies, the tableau algorithm presented here is of the same worst case complexity as for $\mathcal{SHIQ}$, namely 2NExpTime. A detailed investigation of the exact complexity will be part of future work.

3.5 Evaluation of the $\mathcal{RIQ}$ algorithm in FaCT

In order to evaluate the practicability of the above algorithm, we have extended the DL system FaCT [30] to deal with $\mathcal{RIQ}$, and we have carried out a preliminary empirical evaluation.

From a practical point of view, one potential problem with the $\mathcal{RIQ}$ algorithm is that the number of states of automata, and hence the number of different $\forall \mathcal{B}.C$ concepts, could be very large. Moreover, many of these automata could be equivalent (i.e., accept

the same languages). As blocking depends on finding ancestor nodes labelled with the same set of concepts, the discovery of blocks could be unnecessarily delayed, and this can lead to a serious degradation in performance [32].

The FaCT implementation addresses these possible problems by transforming all of the initial NFAs into minimal deterministic finite automata (DFAs), using the AT&T FSM Library™ for this purpose [51]. A minimal DFA is constructed for each role, the states in each DFA are uniquely numbered, and the implementation uses concepts of the form $\forall\mathcal{B}.C$, where $\mathcal{B}$ is the number of a state in one of the DFAs. Determinising the automata allows standard minimisation techniques to be used [28], and because the automata are minimal, if $\forall\mathcal{B}.C$ leads to the presence of $\forall\mathcal{B}'.C$ in some successor node (as a result of repeated applications of the $\forall_2$-rule), then $\forall\mathcal{B}.C$ is equivalent to $\forall\mathcal{B}'.C$ iff $\mathcal{B} = \mathcal{B}'$ (and as $\mathcal{B}$ and $\mathcal{B}'$ are numbers, such comparisons are very easy). Unnecessary blocking delays are thus avoided.

The implementation is still at the "beta" stage, but it has been possible to carry out some preliminary tests using the well-known Galen medical terminology KB [62, 30]. This KB contains 2,740 named concepts and 413 roles, 26 of which are transitive. The roles are arranged in a relatively complex hierarchy with a maximum depth of 10. Classifying this KB using FaCT's $\mathcal{SHIQ}$ reasoner takes 116s on an 800 MHz Pentium III equipped Linux PC. Classifying the same KB using the new $\mathcal{RIQ}$ reasoner took a total of 275s on the same machine. This result is encouraging as it shows that, in the case of the Galen KB at least, using automata in $\forall\mathcal{B}.C$ concepts does not lead to a serious degradation in performance. Moreover, the time taken by the $\mathcal{RIQ}$ reasoner includes approximately 100s to compute the minimal deterministic automata for the role box. This overhead could become important if optimisations of the $\mathcal{RIQ}$ reasoner result in even better performance, but it should be noted that (a) this is a preprocessing step that will not need to be repeated when the remainder of the KB is extended, modified or queried, and (b) compared to other KBs we have seen, the Galen KB involves an unusually large and complex role box.

The KB was then extended with several role inclusion axioms that express the propagation of location across various partonomic roles. These included

$$\texttt{hasLocation}\,\texttt{isSolidDivisionOf} \dot{\sqsubseteq} \texttt{hasLocation}$$

and

$$\texttt{hasLocation}\,\texttt{isLayerOf} \dot{\sqsubseteq} \texttt{hasLocation}.$$

Classifying the extended KB took 280s, an increase of only 2% (3.5% if we exclude the NFA computation time). Subsumption queries w.r.t. this KB revealed that, e.g.,

$$\texttt{Fracture} \sqcap \exists \texttt{hasLocation.NeckOfFemur}$$

was implicitly a kind of

$$\texttt{Fracture} \sqcap \exists \texttt{hasLocation.Femur}$$

(NeckOfFemur is a solid division of Femur), and

$$\texttt{Ulcer} \sqcap \exists \texttt{hasLocation.GastricMucosa}$$

was implicitly a kind of

$$\texttt{Ulcer} \sqcap \exists \texttt{hasLocation}.\texttt{Stomach}$$

(`GastricMucosa` is a layer of `Stomach`). None of these subsumption relationships held w.r.t. the original KB. The times taken to compute these relationships w.r.t. the classified KB could not be measured accurately as they were of the same order as a system clock tick (10ms).

3.6 Summary and Outlook

In this section we have presented an extension of of the well-known expressive DL, $\mathcal{SHIQ}$, with RIAs of the form $RS \dot{\sqsubseteq} P$. We have shown that this extension is undecidable even when RIAs are restricted to the forms $RS \dot{\sqsubseteq} R$ or $SR \dot{\sqsubseteq} R$, but that decidability can be regained by further restricting sets of RIAs to *regular* ones. In the presence of inverse roles, this is slightly tricky, and is realised here using a partial order on role names to prevent cyclic dependencies between roles. The definition of regular sets of RIAs aimed at being as general as possible, and still allows for RIAs of the form $RS \dot{\sqsubseteq} S$, $SR \dot{\sqsubseteq} S$, $SS \dot{\sqsubseteq} S$, and $R^- \dot{\sqsubseteq} R$.

We have presented a tableau algorithm for this DL and reported on its implementation in the FaCT system. A preliminary evaluation suggests that the algorithm will perform well in realistic applications and demonstrates that it can provide important additional functionality in a medical terminology application.

Given that $\mathcal{SHIQ}$ is the basis of the OWL ontology language, this extension to $\mathcal{SHIQ}$ could be used as the foundation for a similar extension to OWL. Although this extension is not able to capture all interesting cases (e.g., it cannot capture the “uncle” example), it can be seen to address many of the most common cases, and it has the great benefit that both decidability and empirical tractability are retained.

4 A Datatype Predicate Extension to OWL

4.1 Background and Motivation

In this section, we present an OWL compatible revision of the datatype group approach first presented in [54], in order to extend OWL datatyping with datatype predicates.

Definition 21 (Datatype Predicate) *A* datatype predicate *(or simply* predicate*) p is characterised by an arity $a(p)$, and a predicate extension (or simply* extension*) $E(p)$.* ◇

Here are some examples of predicates:

1. *integer* is a predicate with arity $a(integer) = 1$ and predicate extension $E(integer) = V(integer)$, where $V(integer)$ is the value space of *integer*. In general, datatypes can be seen as predicates with arity 1 and predicate extensions equal to their value spaces.
2. $>^{int}_{[18]}$ is a unary predicate, with $a(>^{int}_{[18]}) = 1$ and $E(>^{int}_{[18]}) = \{i \in E(integer) \mid i > 18\}$. We can use $>^{int}_{[18]}$ represent a derived XML Schema datatype derived from xsd:integer, with 18 as the value of the minExclusive facet.
3. $=^{int}$ is a binary predicate with arity $a(=^{int}) = 2$ and extension $E(=^{int}) = \{\langle i_1, i_2\rangle \in E(integer)^2 \mid i_1 = i_2\}$.
4. *sum* is a predicate that does not have a fixed arity, where $E(sum) = \{\langle i_1, \ldots, i_n\rangle \in E(integer)^n \mid i_1 = i_2 + \cdots + i_n\}$ and $a(sum) \geq 3$.

In stating the semantics, we assume that datatype interpretations are relativised to a predicate map.

Definition 22 (Predicate Map) *We consider a predicate map $\mathbf{M}_p$ that is a partial mapping from predicate URI references to predicates.* ◇

Example 1 $\mathbf{M}_{p1} = \{\langle$xsd:string$, string\rangle, \langle$xsd:integer$, integer\rangle, \langle$owlx:integerEquality$, =^{int}\rangle, \langle$owlx:integerLargerThanx&n$, >^{int}_{[n]}\rangle\}$ *is a predicate map, where* xsd:string, xsd:integer, owlx:integerEquality *and* owlx:integerLargerThanx&n *are predicate URI references, string, integer and $>^{int}_{[n]}$ are unary predicates, and $=^{int}$ is a binary predicate. Note that, by '$>^{int}_{[n]}$', we mean there exist a predicate $>^{int}_{[n]}$ for each integer n, which is represented by the predicate URI* owlx:integerLargerThanx&n. ♢

Similar to supported and unsupported datatype URIs, we have supported and unsupported predicate URIs according to a predicate map.

Definition 23 (Supported and Unsupported Predicate URIs) *Given a predicate map $\mathbf{M}_p$, a predicate URI u is called a* supported predicate URI w.r.t. $\mathbf{M}_p$ *(or simply* supported predicate URI*), if there exists a predicate p s.t. $\mathbf{M}_p(u) = p$ (in this case, p is called a* supported predicate *w.r.t. $\mathbf{M}_p$); otherwise, u is called an* unsupported predicate URI w.r.t. $\mathbf{M}_p$ *(or simply* unsupported predicate URI*).* ◇

E.g., owlx:integerEquality is a supported predicate URI w.r.t. $\mathbf{M}_{p_1}$ presented in Example 1, while owlx:integerInequality is an unsupported predicate URI w.r.t. $\mathbf{M}_{p_1}$. Therefore, according to $\mathbf{M}_{p_1}$, we know neither the arity nor the extension of the predicate that owlx:integerInequality represents. Note that we make as few as assumptions as possible about unsupported predicates; e.g., we do not even assume that they have a fixed arity.

4.1.1 Datatype Groups

Informally speaking, a datatype group is a group of supported predicate URIs ('wrapped' around a set of base datatype URIs), which can potentially be divided into different sub-groups, so that predicates in each sub-group are about the base datatype of the sub-group. This allows us to make use of known decidability results about the satisfiability problems of predicate conjunctions of, e.g., the admissible/computable concrete domains presented in Section 2.4 of [49]. Formally, a datatype group is defined as follows, and the sub-groups are defined in Definition 27.

Definition 24 (Datatype Group) *A* datatype group $\mathcal{G}$ *is a tuple (*$\mathbf{M}_p$,$\mathbf{D}_\mathcal{G}$,dom*), where* $\mathbf{M}_p$ *is the* predicate map *of* $\mathcal{G}$, $\mathbf{D}_\mathcal{G}$ *is the set of* base datatype *URI references of* $\mathcal{G}$, *and* dom *is the* declared domain function *of* $\mathcal{G}$.

We call $\Phi_\mathcal{G}$ *the set of supported predicate URI references of* $\mathcal{G}$, *i.e., for each* $u \in \Phi_\mathcal{G}$, $\mathbf{M}_p(u)$ *is defined; we require* $\mathbf{D}_\mathcal{G} \subseteq \Phi_\mathcal{G}$. *We assume that there exists a unary predicate URI reference* owlx:DatatypeBottom $\notin \Phi_\mathcal{G}$.

The declared domain function dom *is a mapping s.t.* $\forall u \in \mathbf{D}_\mathcal{G}$: $\mathsf{dom}(u) = u$, *and* $\forall u \in \Phi_\mathcal{G}$, $\mathsf{dom}(u) \in (\mathbf{D}_\mathcal{G})^n$, *where* $n = a(\mathbf{M}_p(u))$. $\diamond$

As we can see from the above definition, supported predicate URIs in $\mathbf{D}_\mathcal{G}$ are also treated as base datatype URIs, therefore they can be used in typed literals.[17] Supported predicate URIs relate to base datatypes URIs via the declared domain function dom, which also helps in defining the interpretation of the relativised negated predicate URIs in Definition 25.

Example 2 $\mathcal{G}_1 = (\mathbf{M}_{p_1}, \mathbf{D}_{\mathcal{G}_1}, \mathsf{dom}_1)$ *is a datatype group, where* $\mathbf{M}_{p_1}$ *is defined in Example 1,* $\mathbf{D}_{\mathcal{G}_1} = \{$xsd:string, xsd:integer$\}$, *and* $\mathsf{dom}_1 = \{\langle$xsd:string, xsd:string$\rangle$, $\langle$xsd:integer, xsd:integer$\rangle$, $\langle$owlx:integerEquality, (xsd:integer, xsd:integer)$\rangle$, $\langle$owlx:*integerLargerThanx&n*, xsd:integer$\rangle\}$.

According to $\mathbf{M}_{p_1}$, *we have* $\Phi_{\mathcal{G}_1} = \{$xsd:string, xsd:integer, owlx:integerEquality, owlx:integer*LargerThanx&n*$\}$. $\diamondsuit$

Definition 25 (Interpretation of Datatype Group) *A* datatype interpretation $I_\mathbf{D}$ *of a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *is a pair* $(\Delta_\mathbf{D}, \cdot^\mathbf{D})$, *where* $\Delta_\mathbf{D}$ *(the datatype domain) is a non-empty set and* $\cdot^\mathbf{D}$ *is a datatype interpretation function, which has to satisfy the following conditions*

1. rdfs:Literal$^\mathbf{D} = \Delta_\mathbf{D}$;

[17] Typed literals are of the form "*v*"^^*u*, where *v* is a lexical form of a data value and *u* is a datatype URI.

2. *for each plain literal l, $l^{\mathbf{D}} = l \in \mathbf{PL}$, where $\mathbf{PL}$ is the value space for plain literals (i.e., the union of the set of Unicode strings and the set of pairs of Unicode strings and language tags);*

3. $\forall u \in \mathbf{D}_{\mathcal{G}}$, *let* $d = \mathbf{M}_p(u)$:

   (a) $u^{\mathbf{D}} = V(d) \subseteq \Delta_{\mathbf{D}}$,

   (b) *if* $v \in L(d)$, *then* $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathbf{D}} = L2V(d)(v)$,

   (c) *if* $v \notin L(d)$, *then* $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathbf{D}}$ *is not defined;*

4. *for any two* $u_1, u_2 \in \mathbf{D}_{\mathcal{G}}$: $u_1^{\mathbf{D}} \cap u_2^{\mathbf{D}} = \emptyset$;

5. $\mathbf{PL} \subseteq \Delta_{\mathbf{D}}$, *and* $\forall u \in \mathbf{D}_{\mathcal{G}}, u^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$;

6. owlx:DatatypeBottom$^{\mathbf{D}} = \emptyset$;

7. $\forall u \in \Phi_{\mathcal{G}}$, $u^{\mathbf{D}} = E(\mathbf{M}_p(u))$;

8. $\forall u \in \Phi_{\mathcal{G}}$, $u^{\mathbf{D}} \subseteq (\mathsf{dom}(u))^{\mathbf{D}}$, *where* $(\mathsf{dom}(u))^{\mathbf{D}} = d_1^{\mathbf{D}} \times \cdots \times d_n^{\mathbf{D}}$ *for* $\mathsf{dom}(u) = (d_1, \ldots, d_n)$ *and* $a(\mathbf{M}_p(u)) = n$.

9. $\forall u \notin \Phi_{\mathcal{G}}$, $u^{\mathbf{D}} \subseteq \bigcup_{n \geq 1} (\Delta_{\mathbf{D}})^n$, *and* $\text{“}v\text{”}\hat{}\hat{}u \in \Delta_{\mathbf{D}}$.

Moreover, we extend $\cdot^{\mathbf{D}}$ *to (relativised) negated predicate URI references* $\overline{u}$ *as follows:*

$$(\overline{u})^{\mathbf{D}} = \begin{cases} \Delta_{\mathbf{D}} \setminus u^{\mathbf{D}} & \text{if } u \in \mathbf{D}_{\mathcal{G}} \\ (\mathsf{dom}(u))^{\mathbf{D}} \setminus u^{\mathbf{D}} & \text{if } u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}} \\ \bigcup_{n \geq 1} (\Delta_{\mathbf{D}})^n \setminus u^{\mathbf{D}} & \text{if } u \notin \Phi_{\mathcal{G}}. \end{cases}$$

◇

Condition 4 requires the value spaces of the base datatype are disjoint, which is essential to dividing $\Phi_{\mathcal{G}}$ into sub-groups. Condition 5 states that the union of the value spaces of plain literals and base datatypes is a proper subset of the datatype domain, because a typed literal associated with an unsupported predicate can be interpreted as something outside the above value spaces. Condition 6 states that owlx:DatatypeBottom is a negated predicate URI of rdfs:Literal. Condition 7 and 8 ensure that the supported predicate URIs are interpreted as the extensions of the predicates they represent, and are subsets of the corresponding declared domains. Condition 9 ensures that unsupported predicate URIs are not restricted to any fixed arity, and that typed literals with unsupported predicates are interpreted as some member of the datatype domain.

Note that supported predicate URIs $u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}}$ have relativised negations (to their declared domains). E.g., $\overline{\text{owlx:integerLargerThanx\&18}}$, the negated predicate URI for owlx:integerLargerThanx&18, is interpreted as $V(\textit{integer}) \setminus (\text{owlx:integerLargerThan}x\&18)^{\mathbf{D}}$; therefore, its interpretation includes the integer 5, but not the string “Fred”, no matter if there exist any other base datatypes in $\mathbf{D}_{\mathcal{G}}$.

Now we introduce the kind of basic reasoning mechanisms required in a datatype group.

Definition 26 (Predicate Conjunction) *Let* $\mathbf{V}$ *be a set of variables,* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *a datatype group, we consider predicate conjunctions of* $\mathcal{G}$ *of the form*

$$\mathcal{C} = \bigwedge_{j=1}^{k} w_j(v_1^{(j)}, \ldots, v_{n_j}^{(j)}), \tag{2}$$

where the $v_i^{(j)}$ *are variables from* $\mathbf{V}$, w_j *are (possibly negated) predicate URI references of the form* u_j *or* $\overline{u_j}$, *and if* $u_j \in \Phi_\mathcal{G}, a(\mathbf{M}_p(u_j)) = n_j$. *A predicate conjunction* $\mathcal{C}$ *is called* satisfiable *iff there exists a function* δ *mapping the variables in* $\mathcal{C}$ *to data values in* $\Delta_\mathbf{D}$ *s.t.* $\langle \delta(v_1^{(j)}), \ldots, \delta(v_{n_j}^{(j)}) \rangle \in w_j^\mathbf{D}$ *for all* $1 \leq j \leq k$. *Such a function* δ *is called a* solution *for* $\mathcal{C}$. $\diamond$

E.g., $\mathcal{C}_1 = \overline{\text{owlx:integerLargerThanx\&38}}(v_1) \wedge \text{owlx:integerLargerThanx\&12}(v_2) \wedge \text{owlx:integerEquality}(v_1, v_2)$ is a predicate conjunction of $\mathcal{G}_1$ presented in Example 2 on page 44. The function $\delta = \{v_1 \mapsto 26, v_2 \mapsto 26\}$ is a solution of $\mathcal{C}_1$; therefore, $\mathcal{C}_1$ is satisfiable.

The predicate conjunction over a datatype group $\mathcal{G}$ can possibly be divided into independent sub-conjunctions of sub-groups of $\mathcal{G}$. Informally speaking, a sub-group includes a base datatype URI and the set of supported predicate URIs about the base datatype URI.

Definition 27 (Sub-Group) *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_\mathcal{G}$, *the* sub-group of w in $\mathcal{G}$, *abbreviated as* $\mathit{sub\text{-}group}(w)$, *is defined as:*

$$\mathit{sub\text{-}group}(w) = \{u | u \in \Phi_\mathcal{G} \text{ and } \mathsf{dom}(u) = \underbrace{(w, \ldots, w)}_{n \text{ times}}\}$$

where $n = a(\mathbf{M}_p(u))$. $\diamond$

Example 3 *The sub-group of* xsd:integer *in* $\mathcal{G}_1$ *presented in Example 2 on page 44 is* $\mathit{sub\text{-}group}(\text{xsd:integer}) = \{\text{xsd:integer}, \text{owlx:integerEquality}, \text{owlx:integerLargerThanx\&}n\}$. *According to the above definition and condition 4 of Definition 25, the predicate conjunction over* $\mathit{sub\text{-}group}(\text{xsd:integer})$ *and* $\mathit{sub\text{-}group}(\text{xsd:string})$ *can be handled separately if there are no common variables; if there are common variables, there exist contradictions, due to the disjointness of* $V(integer)$ *and* $V(string)$. $\diamondsuit$

Since the datatype domain $\Delta_\mathbf{D}$ of a datatype group is not fixed, an admissible concrete domain can no longer be a conforming datatype group (cf. Lemma 4 in [54]). However, a sub-group of a datatype group is very close to a concrete domain; the following definition, accordingly, defines the *corresponding concrete domain* of a sub-group in a datatype group.

Definition 28 (Corresponding Concrete Domain) *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_\mathcal{G}$, *let* $\mathbf{M}_p(w) = \mathcal{D}$, *the* corresponding concrete domain *of* $\mathit{sub\text{-}group}(w)$ *is* $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, *where* $\Delta_\mathcal{D} := V(\mathcal{D})$ *and* $\Phi_\mathcal{D} = \{\bot_\mathcal{D}\} \cup \{\mathbf{M}_p(u) | u \in \mathit{sub\text{-}group}(w)\}$, *where* $\bot_\mathcal{D}$ *corresponds to* $\overline{w}$. $\diamond$

Example 4 *The corresponding concrete domain of sub-group*(xsd:integer) *in* $\mathcal{G}_1$ *presented in Example 2 is* $(\Delta_{integer}, \Phi_{integer})$*, where* $\Delta_{integer} := V(integer)$ *and* $\underline{\Phi_{integer}} = \{\bot_{integer}, integer, =^{int}, >^{int}_{[n]}\}$*. Note that the predicate* $\bot_{integer}$ *corresponds to* xsd:integer*, the negated form of* xsd:integer. ♢

The benefit of introducing the corresponding concrete domain for a sub-group is that if the corresponding concrete domain is admissible, informally speaking, the sub-group is computable.

Lemma 1 *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_\mathcal{G}$*, if the corresponding concrete domain of* w*,* $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$*, is admissible, then the satisfiability problem for finite predicate conjunctions* $\mathcal{C}_w$ *of the* sub-group(w) *is decidable.*

Proof: Direct consequence of Definition 28 and Definition 2.8 on page 28 of [49]: (i) If $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ is admissible, then $\Phi_\mathcal{D}$ is close under negation; hence $\forall u \in \mathsf{sub\text{-}group}(w) \setminus \{w\}$, there exists $u' \in \mathsf{sub\text{-}group}(w)$, such that $\overline{u}^{\mathbf{D}} = u'^{\mathbf{D}}$. Therefore, predicate conjunctions over $\mathsf{sub\text{-}group}(w)$ can be equivalently transformed into predicate conjunctions of $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$. (ii) Predicate conjunctions over $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ are decidable, if $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ is admissible. ∎

Now we provide the conditions for comforming/computable datatype groups.

Definition 29 (Conforming Datatype Group) *A datatype group* $\mathcal{G}$ *is conforming iff*

1. *for any* $u \in \Phi_\mathcal{G} \setminus \mathbf{D}_\mathcal{G}$ *with* $a(\mathbf{M}_p(u)) = n \geq 2$*:* $\mathsf{dom}(u) = (\underbrace{w, \ldots, w}_{n \text{ times}})$ *for some* $w \in \mathbf{D}_\mathcal{G}$*, and*
2. *for any* $u \in \Phi_\mathcal{G} \setminus \mathbf{D}_\mathcal{G}$*: there exist* $u' \in \Phi_\mathcal{G} \setminus \mathbf{D}_\mathcal{G}$ *such that* $u'^{\mathbf{D}} = \overline{u}^{\mathbf{D}}$*, and*
3. *the satisfiability problems for finite predicate conjunctions of each sub-group of* $\mathcal{G}$ *is decidable, and*
4. *for each datatype* $u_i \in \mathbf{D}_\mathcal{G}$*, there exists* $w_i \in \Phi_\mathcal{G}$*, s.t.* $\mathbf{M}_p(w_i) = \neq_{u_i}$ *where* $\neq_{u_i}$ *is the binary inequality predicate for* $\mathbf{M}_p(u_i)$*.* ◇

In the above definition, condition 1 ensure that $\Phi_\mathcal{G}$ can be completely divided into sub-groups. Condition 2 and 3 and all the sub-groups are computable. Condition 4 ensures that number restrictions can be handled.

Example 5 $\mathcal{G}_1$ *presented in Example 2 is not conforming, because it doesn't satisfy condition2 and 4 of the above definition. To make it conforming, we should extend* $\mathbf{M}_{p_1}$ *as follows:* $\mathbf{M}_{p_1} = \{\langle$xsd:string$, string\rangle$, $\langle$owlx:stringEquality$, =^{str}\rangle$, $\langle$owlx:stringInequality$, \neq^{str}\rangle$, $\langle$xsd:integer$, integer\rangle$, $\langle$owlx:integerEquality$, =^{int}\rangle$, $\langle$owlx:integerInequality$, \neq^{int}\rangle$, $\langle$owlx:integerLargerThanx&n$, >^{int}_{[n]}\rangle$, $\langle$owlx:integerLessThanOrEqualx&n$, \leq^{int}_{[n]}\rangle\}$. ♢

Lemma 2 *If $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ is a conforming datatype group, then the satisfiability problem for finite predicate conjunctions of $\mathcal{G}$ is decidable.*

Proof: Let the predicate conjunction be $\mathcal{C} = \mathcal{C}_{w_1} \wedge \cdots \wedge \mathcal{C}_{w_k} \wedge \mathcal{C}_U$,where $\mathbf{D}_\mathcal{G} = \{w_1, \ldots, w_k\}$ and $\mathcal{C}_{w_i}$ is the predicate conjunction for $\mathsf{sub\text{-}group}(w_i)$ and $\mathcal{C}_U$ the sub-conjunction of $\mathcal{C}$ where only unsupported predicate appear.

According to Definition 29, $\mathcal{C}_{w_1} \wedge \cdots \wedge \mathcal{C}_{w_k}$ is decidable. According to Definition 24, $\mathcal{C}_U$ is *unsatisfiable* iff there exist $u(v_1, \ldots, v_n)$ and $\overline{u}(v_1, \ldots, v_n)$ for some $u \notin \Phi_\mathcal{G}$ appear in $\mathcal{C}_U$; otherwise, $\mathcal{C}_U$ is *satisfiable*. Therefore, $\mathcal{C}$ is *satisfiable* iff both $\mathcal{C}_{w_1} \wedge \cdots \wedge \mathcal{C}_{w_k}$ and $\mathcal{C}_U$ are *satisfiable*; otherwise, $\mathcal{C}$ is *unsatisfiable*. ∎

4.1.2 Summary

When we extend OWL datatyping to predicates by datatype groups, we consider the similarities and differences between datatypes and predicates: on the one hand, datatypes can be seen as unary predicates; on the other hand, datatypes are characterised by their lexical spaces, value spaces and lexical-to-value mappings, while predicates are characterised by their arities and extensions. For datatypes, we concern more about their members, i.e., data values; therefore, we could use datatype URI references in typed literals. Predicates are more suitable to represent constraints about data values than datatypes in that they can represent not only unary but also *n*-ary constraints.

In a datatype group, predicates can be divided into some sub-groups, each of which is about a base datatype of the datatype group. The motivations of grouping come from the observation that the predicate conjunction problem of each (some) sub-group(s) is (are) decided by a datatype reasoner. More importantly, the decidability of the predicate conjunction problem of a datatype group depends of the decidability of the sub-problems of all its sub-groups.

Based on the datatype group approach, we propose OWL-E [56], which is a language extending OWL DL with datatype expression axioms, as well as the datatype group-based class constructors to allow the use of datatype expressions in class restrictions. The novelty of OWL-E is that it enhances OWL DL with much more datatype expressiveness and it is still decidable.

4.2 SWRL-P: Extending SWRL with Predicates

This section presents SWRL-P, an extension of SWRL 0.5 (Semantic Web Rule Language, cf. Section2 on page 3) with datatype predicates (or simply *predicates*), based on the OWL predicate extension presented in Section 4.1 on page 43. We will compare SWRL-P and SWRL 0.7 in Section 4.2.3.

SWRL-P extends the set of SWRL atoms to include predicate atoms (or *built-in atoms*);[18] both the abstract syntax and the model-theoretic semantics are extended accordingly. Predicate atoms are of the form $builtin(p, v_1, \ldots, v_n)$, where p is a predicate

[18]We call predicates *built-ins*, following SWRL 0.7, which is available at `http://www.daml.org/rules/proposal/`.

URI reference, and $v_1, \ldots, v_n$ are either literals or variables. Predicate atoms can be used in both the antecedent (body) and consequent (head).

4.2.1 Abstract Syntax

SWRL-P extends axioms to also allow predicate atoms, by adding the production:

atom ::= builtIn '(' dataPredicateID { d-object } ')'

Example 6 *We can define a business rule that one charges no shipping fees for orders (selected items only) over 50 dollars.*

```
Implies(
  Antecedent(priceInDollars(I-variable(x1) D-variable(t1)),
             SelectedItems(I-variable(x1)),
             builtIn(owl:integerGreaterThan
                     D-variable(t1), "50"^^xsd:integer))
  Consequent(shippingFeeInDollars(I-variable(x1) "0"^^xsd:integer))
)
```

In human readable syntax, this rule can be written as

$priceInDollars(?x1, ?t1) \wedge SelectedItems(?x1) \wedge$ owl:integerGreaterThan$(?t1, \text{“50”}\hat{}\hat{}xsd{:}integer)$
$\rightarrow$ $shippingFeeInDollars(?x1, \text{“0”}\hat{}\hat{}xsd{:}integer))$ ♢

4.2.2 Direct Model Theoretic Semantics

Given a datatype group $\mathcal{G}$, We extend an OWL interpretation to a tuple of the form

$$I_p = \{\mathbf{R}, EC, ER, EP, L, S, \mathbf{LV}\}$$

where **R** is a set of resources, $\mathbf{LV} \subseteq \mathbf{R}$ is a set of literal values (the datatype domain of $\mathcal{G}$), EC is a mapping from class descriptions to subsets of **R**, ER is a mapping from property URIs to binary relations on R, EP is a mapping from supported predicate URIs $u \in \Phi_{\mathcal{G}}$ to the predicate extensions $E(\mathbf{M}_p(u))$ of the predicates they represent[19] and from unsupported predicate URIs $u \notin \Phi_{\mathcal{G}}$ to subsets of $\bigcup_{n \geq 1}(\mathbf{LV})^{\mathbf{n}}$, L is a mapping from typed literals to elements of **LV**, and S is a mapping from individual names to elements of EC(owl:Thing).

Given a datatype group $\mathcal{G}$ and an extended abstract OWL interpretation I_p, a binding B(I_p) is an extended abstract OWL interpretation that extends I_p such that S maps i-variables to elements of EC(owl:Thing) and L maps d-variables to elements of **LV** respectively. An atom is satisfied by an interpretation I_p under the conditions given in the Interpretation Conditions Table 2, where C is an OWL DL class description, P is an OWL DL individualvalued property URI, Q is an OWL DL datavalued property URI, u is a predicate URI, x, y are variables or OWL individual URIs, and z, $z_1, \ldots, z_n$ are variables or typed literals.

A binding B(I_p) satisfies an antecedent A iff A is empty or B(I_p) satisfies every atom in A. A binding B(I_p) satisfies a consequent C iff C is not empty and B(I_p) satisfies every

[19] cf. Definition 21.

Atom	Condition on Interpretation
$\mathsf{C}(x)$	$S(x) \in EC(\mathsf{C})$
$P(x,y)$	$\langle S(x), S(y)\rangle \in ER(P)$
$Q(x,z)$	$\langle S(x), L(z)\rangle \in ER(Q)$
$u(z_1,\ldots,z_n)$	$\langle L(z_1),\ldots,L(z_n)\rangle \in EP(u)$
$sameAs(x,y)$	$S(x) = S(y)$
$dofferemtFrom(x,y)$	$S(x) \neq S(y)$

Table 2: Interpretation Conditions Table

atom in C. A rule is satisfied by an interpretation I_p iff for every binding B such that B(I_p) satisfies the antecedent, B(I_p) also satisfies the consequent.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology O_2 is entailed by an ontology O_1 iff every interpretation that satisfies O_1 also satisfies O_2.

Example Consider, for example, the "shipping fee" rule from Section 4.2.1. Assuming that *priceIn Dollars* and *shippingFeeInDollars* are datavaluedPropertyIDs, SeletedItems is a description, and owlx:integerGreaterThan is a predicate URI, then given an interpretation $I = \langle R, EC, ER, EP, L, S, LV\rangle$, a binding $B(I)$ extends S to map the variable $?x_1$ to an element of EC(owl:Thing) and extends L to map the variable $?t_1$ to a data value in **LV**; we will use x_1 to denote the element and t_1 to denote the data value. The antecedent of the rule is satisfied by $B(I)$ iff $(x_1,t_1) \in ER(priceInDollars)$, $x_1 \in EC(\mathsf{SeletedItems})$ and $(t_1, L2V(integer)(\text{“50”}\text{^^}\text{xsd:integer})) \in EP(\text{owlx:integerGreaterThan})$, where $L2V(integer)$ is the lexical-to-value mapping of *integer*. The consequent of the rule is satisfied by $B(I)$ iff $(x_1, L2V(integer)(\text{“0”}\text{^^}\text{xsd:integer})) \in ER(shippingFeeInDo\text{-}llars)$.

Thus the rule is satisfied by I iff for every binding $B(I)$ such that $(x_1,t_1) \in ER(price\ \ InDollars)$, $x_1 \in EC(\mathsf{SeletedItems})$ and $(t_1, L2V(integer)(\text{“50”}\text{^^}\text{xsd:integer})) \in EP$ (owlx:integerGreaterThan), then it is also the case that $(x_1, L2V(integer)(\text{“0”}\text{^^}\text{xsd:integer})) \in ER(shippingFeeInDollars)$, i.e.:

$$\begin{aligned}
&\forall x_1 \in EC(\text{owl:Thing}), t_1 \in \mathbf{LV}.\\
&\quad ((x_1,t_1) \in ER(priceInDollars) \wedge x_1 \in EC(\mathsf{SeletedItems}) \wedge\\
&\qquad (t_1, L2V(integer)(\text{“50”}\text{^^}\text{xsd:integer})) \in EP(\text{owlx:integerGreaterThan}))\\
&\rightarrow (x_1, L2V(integer)(\text{“50”}\text{^^}\text{xsd:integer})) \in ER(shippingFeeInDollars)
\end{aligned}$$

4.2.3 SWRL-P vs. SWRL 0.7

In this section, we briefly compare the SWRL-P and SWRL 0.7.[20] SWRL-P follows the syntax of SWRL 0.7, except that SWRL-P allows the use of unsupported predicate URI references as dataPredicateID; in this sense, SWRL-P is closer to the OWL datatyping.

[20] cf. `http://www.daml.org/rules/proposal/`.

SWRL 0.7 is based on a naive extension of OWL datatyping. It does not distinguish datatypes from predicates, such that it is not clear whether predicates or builtins can be used with typed literal or not in SWRL 0.7. Furthermore, it does not consider the semantics of negated predicate URIs.

5 Other Proposed Extensions

The standardisation of OWL, and its widespread adoption as the knowledge representation language of choice, has motivated research into a wide range of extensions addressing the needs of various different applications.

5.1 Alternative Semantics for OWL Rules

As we have seen in Section 2, although the Horn rules extension proposed in ORL/SWRL has the advantage of simplicity and a tight integration with the existing OWL language, it has the disadvantage that key reasoning problems are no longer decidable. This is a rather serious disadvantage: maintaining the decidability of these problems was an important requirement in the design of OWL, and the loss of decidability indicates that developing effective implementations is likely to be much more problematical.

These considerations (amongst others) have led several groups to study alternative ways of integrating rules with (the DLs underlying) OWL without losing decidability. One obvious way to do this is to restrict (syntactically) the form of rules. A number of different groups have studied this approach, with perhaps the best known work being by Levy et al in the development of the CARIN system [47]. More recently, Calvanese et al have proposed a relatively weak conceptual knowledge representation language that can be combined with rules in such a way that logical implication for ground literals (i.e., individuals) is still decidable, and has relatively low complexity (i.e., polynomial in data complexity) [16]. The main difficulty with both these approaches is that the concept language supported is much weaker than OWL (even OWL Lite), and in the case of CARIN the integration between rules and the concept language is also quite weak—it would not, for example, be possible to use the rules language to capture the “uncle” property discussed in Section 2.

An alternative approach is to weaken the semantic connection between rules and the concept language. This can be done by restricting the effect of rules to the Herbrand universe, i.e., to individuals named in the ontology. This approach has a long history in Description Logics, and was, for example, used in the Classic system [11]. More recently, variations on this approach have been studied with a view to providing a decidable solution for OWL rules. Of particular interest is work by Eiter et al on the combination of answer set programming with DLs [20]. This would provide for a decidable language, but would have the disadvantage that rule based inferences would only affect inferences relating to individuals (e.g., retrieval queries), and would not affect inferences relating to classes (e.g., subsumption).

In a similar vein, an autoepstemic semantics can be applied to rules as first proposed by Donini et al [19], and more recently investigated in the context of OWL by Franconi et al [23]. In this approach is used to limit the effects of rules to inferences relating to individuals. The decidability of key reasoning tasks in the resulting language can be demonstrated, but their precise complexity bounds are still unknown.

5.2 A Fuzzy Extension to OWL

In some applications it is important to be able to deal with uncertain, imprecise and/or vague knowledge. This requirement has led to a recent investigation by Kollias et al of a "fuzzy" extension to both OWL and SWRL [77]. This approach is based on the "fuzzification" of the interpretation, and does not change the syntax of the concept and role constructors.

This is not the first time that fuzzy extensions to description logics have been studied. Earlier work by Yen [79] and by Tresp and Molitor [75] also suggested leaving the basic DL syntax unchanged, although Tresp and Molitor proposed an extension to allow a fuzzy membership value to be applied to concepts.

5.3 A Context Extension to OWL

Some researchers have argued that OWL needs to be extended to better meet the requirements arising from the integrating multiple heterogeneous ontologies [13]. In particular, it is argued that interpreting all ontologies in the same domain is unsuited to such an integration scenario.

The proposed solution is to allow for multiple interpretation domains, with ontologies in different domains being linked by *bridging rules*. These bridging rules effectively establish subsumption relationships between classes in different interpretation domains. Note that this approach is rather similar in many respects to the work by Walter et al on E-connections [45].

It is claimed that the approach can be adapted not only to OWL but also to SWRL and to other extensions of OWL (such as the fuzzy extension mentioned above).

6 Conclusion

As we have seen, the importance of ontologies in the Semantic Web has prompted the development of several proposed extensions to the OWL ontology language. These include extensions for rules, fuzzy concepts, datatypes and multiple contexts.

In this report we have described in detail three of the more prominent and well developed proposals. It is likely that the ORL/SWRL proposal will lead to the establishment of a new W3C standardisation working group with a view to developing a new standard for Semantic Web rules. The approach using complex role inclusion axioms, while attractive in some respects (i.e., the retention of decidability) does not seem so likely to be adopted as it does not offer the same increase in expressive power, and it seems to be already widely accepted that future extensions to OWL will no longer be decidable. It seems likely that the proposal to extend OWL with more expressive datatypes will be merged into the SWRL proposal, and more recent versions of this proposal include a wide range of built in datatype predicates (see [37]).

Other extensions to OWL and its underlying DL, such as those described in Section 5, are generally less well developed, and seem much less likely to find their way into language standardisation proposals in the short term. Some of this work is, however, very promising and might soon begin to have an impact on the development of languages for the Semantic Web.

References

[1] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence (AAAI'90)*, pages 621–626, Boston (Ma, USA), 1990.

[2] Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *J. of Logic, Language and Information*, 2:1–18, 1993.

[3] Franz Baader and Ulrike Sattler. Number restrictions on complex roles in description logics: A preliminary report. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 328–338, 1996.

[4] M. Baldoni. *Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension*. PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy, 1998.

[5] M. Baldoni, L. Giordano, and A. Martelli. A tableau calculus for multimodal logics and some (un)decidability results. volume 1397 of *Lecture Notes in Artificial Intelligence*. Springer, 1998.

[6] Dave Beckett. Rdf/xml syntax specification (revised). W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-syntax-grammar/`.

[7] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[8] R. Berger. The undecidability of the dominoe problem. 66, 1966.

[9] Tim Berners-Lee. Semantic web roadmap, 1998. Available at `http://www.w3.org/DesignIssues/Semantic`.

[10] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001. Available at `http://www.w3.org/TR/xmlschema-2/`.

[11] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[12] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[13] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, pages 164–179. Springer, 2003.

[14] R. J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[15] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler. Extensible markup language (XML) 1.0 (second edition). W3C recommendation., October 2000. Available at `http://www.w3.org/TR/1998/REC-xml`.

[16] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. What to ask to a peer: Ontolgoy-based query reformulation. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 469–478. Morgan Kaufmann, Los Altos, 2004.

[17] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics (extended abstract). In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*. AAAI Press, 1994.

[18] S. Demri. The complexity of regularity in grammar logics and related modal logics. *J. of Logic and Computation*, 11(6), 2001.

[19] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Adding epistemic operators to concept languages. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 342–353. Morgan Kaufmann, Los Altos, 1992.

[20] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 141–151. Morgan Kaufmann, Los Altos, 2004.

[21] L. Farinãs del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121-122:123–134, 1988.

[22] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[23] Enrico Franconi, Gabriel M. Kuper, Andrei Lopatenko, and Luciano Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *International VLDB Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03)*, pages 64–76, 2003.

[24] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.

[25] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.

[26] Patrick Hayes. RDF model theory. W3C Recommendation, 10 February 2004. Available at http://www.w3.org/TR/rdf-mt/.

[27] Laura Hollink, Guus Schreiber, Jan Wielemaker, and Bob Wielinga. Semantic annotation of image collections. In *Workshop on Knowledge Markup and Semantic Annotation, KCAP'03*, 2003. Available at http://www.cs.vu.nl/~guus/papers/Hollink03c.pdf.

[28] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison Wesley Publ. Co., Reading, Massachussetts, 1997.

[29] Masahiro Hori, Jérôme Euzenat, and Peter F. Patel-Schneider. OWL web ontology language XML presentation syntax. W3C Note, 11 June 2003. Available at http://www.w3.org/TR/owl-xmlsyntax/.

[30] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*. Morgan Kaufmann, Los Altos, 1998.

[31] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. Technical Report LTCS-Report 02-06, TU-Dresden, Germany, 2002.

[32] I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th Eur. Conf. on Artificial Intelligence (ECAI 2002)*, 2002.

[33] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*. Morgan Kaufmann, Los Altos, 2003. A long version is available as technical report LTCS 02-06 at http://lat.inf.tu-dresden.de/research/reports.html.

[34] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 161–180. Springer, 1999.

[35] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 17–29. Springer, 2003.

[36] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an owl rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

[37] 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 Note, 21 May 2004. Available at http://www.w3.org/Submission/SWRL/.

[38] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, pages 792–797. AAAI Press, 2002.

[39] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[40] Ian Horrocks and Ulrike Sattler. The effect of adding complex role inclusion axioms in description logics. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 343–348. Morgan Kaufmann, Los Altos, 2003.

[41] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.

[42] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.

[43] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[44] O. Kupferman, U. Sattler, and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proc. of the 19th Int. Conf. on Automated Deduction (CADE 2002)*, volume 2392 of *Lecture Notes in Artificial Intelligence*. Springer, 2002.

[45] Oliver Kutz, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 151(1):1–73, 2004.

[46] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems*. Addison-Wesley, 1989.

[47] Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[48] John W. Lloyd. *Foundations of logic programming (second, extended edition)*. Springer series in symbolic computation. Springer-Verlag, New York, 1987.

[49] Carsten Lutz. Interval-based temporal reasoning with general TBoxes. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 89–94, 2001.

[50] Drew V. McDermott and Dejing Dou. Representing disjunction and quantifiers in rdf. In *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2002.

[51] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. *A Rational Design for a Weighted Finite-State Transducer Library*. Number 1436 in LNCS. Springer, 1998.

[52] Lin Padgham and Patrick Lambrix. A framework for part-of hierarchies in terminological logics. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 485–496, 1994.

[53] Jeff Pan and Ian Horrocks. Web ontology reasoning with datatype groups. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 47–63. Springer, 2003.

[54] Jeff Pan and Ian Horrocks. Web ontology reasoning with datatype groups. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 47–63. Springer, 2003.

[55] Jeff Z. Pan and Ian Horrocks. Extending Datatype Support in Web Ontology Reasoning. In *Proc. of the 2002 Int. Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2002)*, number 2519 in Lecture Notes in Computer Science, pages 1067–1081. Springer, 2002.

[56] Jeff Z. Pan and Ian Horrocks. Owl-e: Extending owl dl with datatype expressions. Technical report, Information Management Group, Computer Science Department, The University of Manchester, April 2004.

[57] P. F. Patel-Schneider and I. Horrocks. DLP and FaCT. volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 19–23. Springer, 1999.

[58] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[59] Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bull.*, 2(3):108–113, 1991.

[60] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. CEUR (`http://ceur-ws.org/`), 2002.

[61] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

[62] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, Menlo Park, California, 1997.

[63] Alan Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proc. of DL 2002*. CEUR (`http://ceur-ws.org/`), 2002.

[64] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[65] U. Sattler. Description logics for the representation of aggregated objects. In W. Horn, editor, *Proc. of the 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)*. IOS Press, Amsterdam, 2000.

[66] Ulrike Sattler. Description logics for the representation of aggregated objects. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)*, 2000.

[67] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[68] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Acta Informatica*, 48(1):1–26, 1991.

[69] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In Ron J. Brachman, Hector J. Levesque, and Ray Reiter, editors, *Proc. of the 1st Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, Los Altos, 1989.

[70] S. Schulz and U. Hahn. Parts, locations, and holes - formal reasoning about anatomical structures. In *Proc. of AIME 2001*, volume 2101 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.

[71] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. OWL web ontology language guide. W3C Recommendation, 10 February 2004. Available at `http://www.w3.org/TR/owl-guide/`.

[72] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed-rt. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.

[73] The DAML Services Coalition. Daml-s: Semantic markup for web services, May 2003. Available at `http://www.daml.org/services/daml-s/0.9/daml-s.html`.

[74] S. Tobies. PSPACE reasoning for graded modal logics. *J. of Logic and Computation*, 11(1):85–106, 2001.

[75] Christopher B. Tresp and Ralf Molitor. A description logic for vague knowledge. In *Proc. of the 13th Eur. Conf. on Artificial Intelligence (ECAI'98)*, pages 361–365, 1998.

[76] Dmitry Tsarkov and Ian Horrocks. DL reasoner vs. first-order prover. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, volume 81 of *CEUR (`http://ceur-ws.org/`)*, pages 152–159, 2003.

[77] V. Tzouvaras, G. Stamou, and S. Kollias. Knowledge refinement using fuzzy compositional neural networks. In *International Conference on Artificial Neural Networks (ICANN'03)*, 2003.

[78] M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. CEUR (`http://ceur-ws.org/`), 2001.

[79] John Yen. Generalizing term subsumption languages to fuzzy logic. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 472–477, 1991.