Integrating OWL and Rules: A Syntax Proposal for Nominal Schemas

David Carral Martínez, Adila A. Krisnadhi, Pascal Hitzler

Kno.e.sis Center, Wright State University, Dayton OH 45435, USA

Abstract. This paper proposes an addition to OWL 2 syntax to incorporate nominal schemas, which is a new description-logic style extension of OWL 2 which was recently proposed, and which makes is possible to express "variable nominal classes" within axioms in an OWL 2 ontology. Nominal schemas make it possible to express DL-safe rules of arbitrary arity within the extended OWL paradigm, hence covering the well-known DL-safe SWRL language. To express this feature, we extend OWL 2 syntax to include necessary and minimal modifications to both Functional and Manchester syntax grammars and mappings from these two syntaxes to Turtle/RDF. We also include several examples to clarify the proposal.

1 Introduction

Nominal schemas [4,5] are a new description-logic style extension of OWL 2 [10] which can be used like "variable nominal classes" within OWL 2 axioms. Although their semantics restricts them only to stand for named individuals, nominal schemas allow us to express arbitrarily shaped (Datalog) rules within the description logic (DL) paradigm, hence pushing the expressivity of OWL 2 DL and its fragments even further.

While the semantic intuition behind nominal schemas is the same as the one behind DL-safe variables presented in [8], the difference lies in the fact that DL-safe variables are tied to rule languages, while nominal schemas integrate seamlessly with DL syntax. The proposed extension encompasses DL-safe variable SWRL [3,9,4] while staying within the DL/OWL language paradigm and without employing hybrid approaches.

Nominal schemas have been introduced as a new general constructor for DL, denoted by the letter $\mathcal{V}$ in the DL nomenclature. The addition of nominal schemas has been considered for several DLs such as $\mathcal{SROIQ}$ that underlies OWL 2 DL, and $\mathcal{SROEL}$ that underlies the OWL 2 EL profile (define DL $\mathcal{SROIQV}$ and $\mathcal{SROELV}$, respectively, as extensions of the DLs $\mathcal{SROIQ}$ and $\mathcal{SROEL}$). It has been shown in [5] that the worst-case complexity of $\mathcal{SROIQV}$ remains N2EXPTIME-complete, i.e., no worse than $\mathcal{SROIQ}$. Furthermore, in the same paper, a tractable fragment of $\mathcal{SROIQV}$ has been identified. This fragment is called $\mathcal{SROELV}_n$ which is obtained by extending $\mathcal{SROEL}$ with

nominal schemas in a slightly restricted form. Nevertheless, it still covers[1] both OWL 2 EL and (DL-safe) OWL 2 RL.

We present an example of nominal schemas in the following. First, rules such as (1) are not expressible in the current OWL 2 DL standard.

$$\text{hasFather}(x,y) \wedge \text{hasBrother}(y,z) \wedge \text{hasTeacher}(x,z) \rightarrow \text{ChildTaughtByUncle}(x) \quad (1)$$

Intuitively, this is due to the fact that the body of the above rules is not tree-shaped. See [4] and [6] (which formally defines not tree-shaped rules that can be expressed in $\mathcal{SROIQ}$ extended with role regularity) for further discussion. In contrast, using nominal schemas, rule (1) can be expressed as (2).

$$\exists \text{hasTeacher}.\{z\} \sqcap \exists \text{hasFather}.\exists \text{hasBrother}.\{z\} \sqsubseteq \text{ChildTaughtByUncle}. \quad (2)$$

The expression $\{z\}$ is a nominal schema, which is to be read as a variable nominal that can only represent nominals (i.e., z binds to known individuals), where the binding is the same for all occurrences of the same nominal schema in an axiom. Variables x and y can still take arbitrary values and are hidden in the DL syntax, z needs to be restricted to be DL-safe to retain the conclusion. For a more detailed description of nominal schemas including their formal semantics see [5].

This document proposes representations for nominal schemas for the prominent variants of OWL syntax: Functional, Manchester, Turtle and RDF/XML. For an introduction of the OWL syntax, consult [10]. Mapping from Turtle triples to RDF/XML is a well defined and automatized process so the RDF/XML based syntax will not be directly addressed in this document, it is assumed that it can be easily derived from the Turtle Syntax.

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

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

The selected approach, the use of a xsd:string datatype, is also considered by the RIF XML format [11]. Note that the same nominal schema can never appear in two different statements of an ontology—more precisely, if the same variable occurs in different axioms, then they are considered distinct (i.e., local to the axiom), in a way similar to the use of variables in rules. So a specific nominal schema is local to one single axiom. By using a string type, the occurrence of

[1] without datatype-related features

the nominal schema is exclusively bound to the axiom where it appears and the same string could be repeated in different axioms along the ontology safely. Even if two nominal schemas use the same string, they will be considered as different occurrences of a datatype and therefore, they can be understood as two separated nodes in an RDF graph.

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

The document is structured as follows. Section 2 contains the necessary modifications that have to be made to the Manchester and Functional Syntax grammars in order to include nominal schemas. Section 3 refers to the mappings from these syntaxes to Turtle. Section 4 concludes. Appendix A contains two examples for the usage of nominal schemas in the different syntax variants that are discussed in the document.

2 Grammar Modifications

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

Functional Syntax Grammar Modifications

We define in this section the required modifications we propose for the Functional Syntax grammar [1]. The reserved word ObjectVariable will be used to mark the appearance of a nominal schema. Nominal schemas will be in parentheses and will always be followed by the expression '^^xsd:string'. The changes are formally defined in the next paragraphs.

Add the next production rule to the grammar:

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

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

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

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

Manchester Syntax Grammar Modifications

Again, the reserved word ObjectVariable will be used to mark the appearance of the nominal schemas in the Manchester Syntax [2]. As in the Functional Syntax, the nominal will be in parentheses and followed by '^^xsd:string'. The needed changes to this grammar are:

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

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

Add the next production rule to the grammar:

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

3 Mapping FS and MS to Turtle

We define the syntax of nominal schemas in Turtle through the mapping from Functional and Manchester Syntaxes to the triple-notation. We assume that from this notation the process to move to RDF/XML is already formalized so, as said before, the XML syntax will not be directly addressed in this document.

Functional Syntax to and from Turtle

The W3C document containing the formal mapping from FS to Turtle can be found in [7]. To add nominal schemas syntax to the mappings, first add the next row to the mapping from FS to Turtle:

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

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

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

Manchester Syntax to and from Turtle

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

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

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

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

4 Conclusions

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

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

References

1. Peter F. Patel-Scheneider Boris Motik and Bijan Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style*. W3C Recommendation 27 October 2009, 2009. http://www.w3.org/TR/owl2-syntax/.
2. Matthew Horridge and Peter F. Patel-Scheneider. *OWL 2 Web Ontology Language: Manchester Syntax*. W3C Working Group Note 27 October 2009, 2009. http://www.w3.org/TR/owl2-manchester-syntax/.
3. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission 21 May 2004, 2004. Available from http://www.w3.org/Submission/SWRL/.
4. Adila A. Krisnadhi, Frederick Maier, and Pascal Hitzler. OWL and Rules. In A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, and P.F. Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data. 7th International Summer School*, pages 382–415. Lecture Notes in Computer Science Vol. 6848, Springer, Heidelberg, 2011, pp. 382-415., Galway, Ireland, August 2011.
5. Markus Krötzsch, Frederick Maier, Adila A. Krisnadhi, and Pascal Hitzler. A Better Uncle for OWL: Nominal Schemas for Integrating Rules and Ontologies. In S. Sadagopan, Krithi Ramamritham, Arun Kumar, M.P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proc. of the 20th International World Wide Web Conference (WWW'11)*, pages 645–654, Hyderabad, India, March/April 2011. ACM, New York.
6. David Carral Martínez and Pascal Hitzler. Extending Description Logic Rules. In *Proceedings of the 9th Extended Semantic Web Conference, ESWC , May 2012.*, Heraklion, Crete, Greece,, 2012. Lecture Notes in Computer Science, Springer, Heidelberg (2012), to appear.
7. Boris Motik and Peter F. Patel-Scheneider, editors. *OWL 2 Web Ontology Language: Mapping to RDF Graphs*. W3C Recommendation 27 October 2009, 2009. http://www.w3.org/TR/owl2-mapping-to-rdf/.
8. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.
9. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. *Web Semant.*, 3:41–60, July 2005.
10. W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.
11. Harold Boley Gary Hallmark Michael Kifer Adrian Paschke Axel Polleres Dave Reynolds, editor. *OWL 2 Web Ontology Language: Manchester Syntax*. W3C Recommendation 22 June 2010, 2010. http://www.w3.org/TR/rif-core.

A Syntax Examples

A.1 Example 1

Rule Syntax

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

DL Syntax

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

Functional Syntax

```
SubClassOf(

ObjectIntersectionOf(
   ObjectSomeValuesFrom( :hasFather
      ObjectSomeValuesFrom( :hasBrother ObjectVariable("v1"^^xsd:string) ))
   ObjectSomeValuesFrom( :hasTeacher ObjectVariable("v1"^^xsd:string) )
)

:ChildTaughtByUncle
)
```

RDF/XML Syntax

```
_:x1 rdfs:subClassOf :ChildTaughtByUncle

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

    _:x2 rdf:type owl:Restriction          _:x3 rdf:type owl:Restriction
    _:x2 owl:onProperty :hasFather         _:x3 owl:onProperty :hasTeacher
    _:x2 owl:someValuesFrom _:x5           _:x3 owl:someValuesFrom _:x4

    _:x4 rdf:type owl:Restriction          _:x6 rdf:type owl:ObjectVariable
    _:x4 owl:onProperty :hasBrother        _:x6 owl:variableId "v1"
    _:x4 owl:someValuesFrom :x6

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

Manchester Syntax

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

A.2 Example 2

Rule Syntax

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

DL Syntax

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

Functional Syntax

```
SubClassOf(

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

:ReviewerWithConflictingAssignment
)
```

RDF/XML Syntax

```
_:x1 rdfs:subClassOf :ReviewerWithConflictingAssignment

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

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

Manchester Syntax

```
Class: ReviewerWithConflictingAssignment
SubtClassOf:
    ( hasReviewAssign some
        ( (hasAuthor some (Variable "v1"^^xsd:string)) and
        (atVenue some (Variable "v2"^^xsd:string)) ) )
    and
    ( :hasSubmittedPaper some
        ( (hasAuthor some (Variable "v1"^^xsd:string))
        and (atVenue some (Variable "v2"^^xsd:string)) ) )
```