A Syntax for Rules in OWL 2

Birte Glimm[1], Matthew Horridge[2], Bijan Parsia[2], Peter F. Patel-Schneider[3]

[1] Oxford University Computing Laboratory, UK
[2] The University of Manchester, UK
[3] Bell Labs, US

Abstract. Being able to extend an OWL ontology with some form of rules is a feature that many ontology developers consider as very important. Nevertheless, working with rules in practice can be difficult since the tool support is not as good as for handling standard ontologies. Furthermore, the existing rule syntaxes are not very well aligned with the new OWL 2 standard. We propose, therefore, an extension to OWL 2 for representing rules, which is directly inspired by (DL Safe) SWRL rules, but uses and extends the succinct and human-readable functional-style syntax of OWL 2. We also propose an OWL/XML version of the syntax for easy XML serialization. Support for parsing such rules has been added to the new OWL API 3.0 and reasoning support is available in the two OWL 2 reasoners Pellet and HermiT. In HermiT, these rules can also be used in conjunction with description graphs.

1 Introduction

In previous OWLED workshops, many ontology developers mentioned rule support as a critical need. Although OWL has several sorts of conditionals, e.g.:

$$\mathsf{SubClassOf(:Person\ ObjectUnionOf(:Human\ :IntelligentComputer))} \tag{1}$$

(if you are a Person, then you are a Human or an IntelligentComputer) these conditionals are very constrained. For example, the SubClassOf conditional can only have class expressions in the "if" or the "then" parts and even in OWL 2 [8], it is not possible to mix classes and properties (directly) as in:

$$\mathsf{SubClassOf(:parentOf\ ObjectUnionOf(:Human\ :IntelligentComputer))} \tag{2}$$

(assuming that parentOf is an object property). The specialized conditionals of OWL have several advantages: they allow for variable-free syntax; they are more intention revealing; and they help enforce restrictions which make OWL easier to process (e.g., by making it decidable). However, this comes at the price of expressivity. Rules are much less restricted and one can easily express the intuition behind axiom (2) with the following rule:

```
Rule( Body(ObjectPropertyAtom(:parentOf Variable(:x) Variable(:y)))
      Head(ClassAtom(ObjectUnionOf(:Human :IntelligentComputer) Variable(:y))))
```

The rule axiom consists of a body (also called antecedent) and a head (also called consequent), each of which consists of a possibly empty set of atoms. Informally, the rule can be read as: if the body is true, then the head must be true. An empty body is trivially true, whereas an empty head is trivially false. In case the head contains several atoms, each atom must hold.

Readers might notice that the above used rule syntax is very close to the functional-style syntax (FSS) of OWL 2 [8] and the resemblance to axioms (1) and (2) is immediate. This is deliberate and the aim for this paper is to propose a rule extension for OWL that is well aligned with the OWL 2 specification. For example, the ObjectPropertyAtom construct can be used similar to the ObjectPropertyAssertion construct of the OWL 2 FSS apart from the fact that it takes variables as well as individual IRIs as arguments.

Although rule support is clearly a desirable feature, there is no general commitment of implementors of OWL reasoners to a particular syntax or semantics, although SWRL rules [7] are the most commonly used ones. The new syntax we propose is directly inspired by (DL Safe) SWRL rules, but uses and extends the functional-style syntax of OWL 2. We also outline OWL/XML, RDF, and Manchester versions of the rule syntax.

Our main reason for proposing this new syntax for rules is to allow rules to be directly integrated into OWL 2 ontologies, permitting a closer integration between rules and core OWL 2 ontologies. This has the added benefit of allowing a better-integrated presentation of rules in ontology editors such as Protege.[4] This new syntax would replace the SWRL syntax. It is, however, not intended as a competitor to RIF [1], as RIF is designed for interchange of rules between different rule systems.

To complete the definition of this rule extension to OWL 2, we provide a semantics for these rules based on standard DL Safe rules. We also provide a version of our rules that can effectively be used with description graphs [4], and show how the semantics of these rules differ from the semantics of DL Safe rules.

Support for parsing such DL Safe rules is being added to the new OWL API 3.0 and reasoning over ontologies with such rules is possible with Pellet[5] and HermiT[6]. Support for parsing ontologies with description graphs and description graph rules is available in an extension of the OWL API[7] and HermiT can be used for reasoning with such ontologies. The tightly coupled syntax together with integration into the widely used OWL API will hopefully lay the foundation for general support of rules in OWL implementations.

2 Preliminaries

We do not give a full introduction into OWL 2 due to space limitations. All examples are assumed to be self-explanatory and further details about OWL 2 are available in the W3C specification [8].

[4] http://protege.stanford.edu/
[5] http://clarkparsia.com/pellet/
[6] http://hermit-reasoner.com/
[7] https://owlapi.svn.sourceforge.net/svnroot/owlapi/v3/branches/owlextensions/

2.1 Rules and Safety Restrictions

Rules, such as SWRL rules and the rules we present in this paper, generalize OWL conditionals in two ways (i) they allow for arbitrary patterns of variables and (ii) they allow for mixing of property and class expressions in a fairly unrestricted way. Unlike many traditional rule languages such as Prolog or Datalog, we adopt the open world assumption, which is used for OWL ontologies and also in SWRL rules.

One problem with rules is that they can easily lead to undecidability if used in an unrestricted way. Thus, we place some restriction on the rules. Firstly, we make the usual "safety" condition that only variables that occur in the body of a rule are allowed to occur in the head of the rule. The rules can still enforce the existence of new individuals by using class atoms in the head with expressions such as ObjectSomeValuesFrom.

We also employ the so-called DL Safe restriction, which requires that individual variables in a rule bind only to individuals named explicitly in the underlying ontology. Without data variables this restriction makes the standard reasoning tasks for ontologies extended with rules decidable. With data variables, one can still have rules that generate infinitely many new inferences. As an example, consider the following rule (for brevity in the also proposed Manchester syntax):

$$\mathsf{Rule:\ Person(?x),\ integer(?y) -> hasPossibleIncome(?x,\ ?y)}$$

This rule would, for any named instance of the class Person generate infinitely many hasPossibleIncome tuples, one for each integer. Restricting the data variables, similarly to individual variables, to data values that occur in the input can result in very counter-intuitive results since, for example, a range of integers greater than 17 and smaller than 19 contains exactly 18, but 18 is never mentioned in the input and can, thus, not be used as a binding with this restriction. We propose, therefore, a different restriction and require that each data variable in a data range atom also occurs in a data property atom in the body. This allows for deferring the rule application until we know about required data valued successors of a named individual. For example, the rule

$$\mathsf{Rule:\ Person(?x),\ hasSSN(?x,\ ?y),\ integer(?y) -> hasID(?x,\ ?y)}$$

satisfies this restriction since the variable ?y from the data range atom integer(?y) also occurs in the data property atom hasSSN(?x, ?y). Such a rule can be rewritten into

$$\mathsf{Rule:\ Person(?x),\ hasSSN(?x,\ ?y) -> not\ integer(?y)\ or\ hasID(?x,\ ?y)}$$

and only when we know that a particular person has some SSN, we can make a non-deterministic decision as to whether ?y is not an integer or a hasID successor of the individual bound to ?x. Standard OWL 2 axioms naturally result in such rules, when translated to First-Order Logic.

2.2 Description Graphs

Similar to rules, description graphs [5] are knowledge modeling constructs that can describe objects with parts connected in arbitrary ways. For example, one can describe benzene as a chemical compound that contains a ring composed of six carbon atoms with alternating double and single bonds (cf. Figure 1).

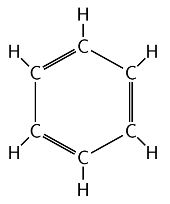

Fig. 1. A representation of the benzene compound.

Since standard DL axioms can only describe tree-like structures, modeling a ring is not directly possible otherwise. Description graphs can also be combined with rules to express conditionals, but since description graphs do not contain named individuals, using the DL Safe restriction is not reasonable. Instead Motik et al. [5] propose a restriction called *strong separation* under which properties from the ontology cannot be mixed with properties used in the description graphs that extend the ontology. For example, if the ontology contains a description graph describing benzene, which uses the properties :doubleBondsTo and :singleBondsTo to connect the carbon atoms (as illustrated by the double and single lines in Figure 1), then a rule can be used to infer that every single bond is a bond if the rule is not applied under the DL Safe restriction. To satisfy the strong separation criterion, it must be possible to separate the properties used in the ontology into two disjoint sets of graph properties and non-graph properties such that graph properties are only used in description graphs and graph rules and non-graph properties are only used in standard OWL 2 axioms and in (DL Safe) non-graph rules. I.e., the properties bondsTo, singleBondsTo, and doubleBondsTo cannot be used in standard OWL 2 axioms and DL Safe rules.

Applying the graph rules without the DL Safe restriction still allows for decidable reasoning because description graphs also have to satisfy an acyclicity condition, which ensures that the description graphs can always be represented by finite structures.

3 Syntax Extension

In this section we describe the new rule syntax that is inspired by the OWL 2 functional-style syntax, the according OWL/XML serialization, and the mapping of rules to RDF graphs.

3.1 Functional-Style Syntax

We specify the rule syntax by means of an extended BNF. Terminal symbols are shown in single quotes and non-terminal symbols are shown in bold face. 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 ({. . .}), and a vertical bar denotes an alternative choice. Whitespace is ignored in the productions given here.

From the OWL 2 Structural Specification and Functional-Style Syntax document [8], we can see that an ontology contains (possibly empty) sets of axioms, annotations, declarations, and import statements. The production rules in Table 1 extend the grammar from the OWL 2 Structural Specification and Functional-Style Syntax (Appendix 13) by allowing for rules as an additional type of axiom.

Table 1. The production rules for the rules and description graph extension in functional-style syntax.

axioms	::=	{ **Axiom** | **Rule** | **DGAxiom** }
Rule	::=	**DLSafeRule** | **DGRule**
DLSafeRule	::=	‘DLSafeRule’ ‘(’ {**Annotation**} ‘Body’ ‘(’ {**Atom**} ‘)’
		‘Head’ ‘(’ {**Atom**} ‘)’ ‘)’
Atom	::=	‘ClassAtom’ ‘(’ **ClassExpression IArg** ‘)’
		| ‘DataRangeAtom’ ‘(’ **DataRange DArg** ‘)’
		| ‘ObjectPropertyAtom’ ‘(’ **ObjectPropertyExpression IArg IArg** ‘)’
		| ‘DataPropertyAtom’ ‘(’ **DataProperty IArg DArg** ‘)’
		| ‘BuiltInAtom’ ‘(’ **IRI DArg** {**DArg**} ‘)’
		| ‘SameIndividualAtom’ ‘(’ **IArg IArg** ‘)’
		| ‘DifferentIndividualsAtom’ ‘(’ **IArg IArg**‘)’
IArg	::=	‘Variable’ ‘(’ **IRI** ‘)’ | **Individual**
DArg	::=	‘Variable’ ‘(’ **IRI** ‘)’ | **Literal**
DGRule	::=	‘DescriptionGraphRule’ ‘(’ {**Annotation**} ‘Body’ ‘(’ {**DGAtom**} ‘)’
		‘Head’ ‘(’ {**DGAtom**} ‘)’ ‘)’
DGAtom	::=	‘ClassAtom’ ‘(’ **ClassExpression IArg** ‘)’
		| ‘ObjectPropertyAtom’ ‘(’ **ObjectPropertyExpression IArg IArg** ‘)’
DGAxiom	::=	‘DescriptionGraph’ ‘(’ {**Annotation**} **DGNodes**
		DGEdges MainClasses‘)’
DGNodes	::=	‘Nodes’‘(’ **NodeAssertion** {**NodeAssertion** } ‘)’
NodeAssertion	::=	‘NodeAssertion’‘(’ **Class DGNode** ‘)’
DGNode	::=	**IRI**
DGEdges	::=	‘Edges’‘(’ **EdgeAssertion** {**EdgeAssertion** } ‘)’
EdgeAssertion	::=	‘EdgeAssertion’ ‘(’ **ObjectProperty DGNode DGNode**‘)’
MainClasses	::=	‘MainClasses’ ‘(’ **Class** {**Class** } ‘)’

In the following section we clarify the semantics of the rules, but it is worth noting that the semantics of built-in atoms are not defined here. It is suggested that implementers who want to add support for built-in atoms look at the SWRL W3C member submission for guidance [7].

3.2 Direct Model-Theoretic Semantics

The model-theoretic semantics is a straightforward extension of the semantics for OWL 2. The interpretation of description graphs is the natural one and we refer interested readers to [5] for further details.

OWL 2 Direct Semantics We give just a short summary of standard OWL 2 semantics [6].

A *datatype map* is a 6-tuple $D = (N_{DT}, N_{LS}, N_{FS}, \cdot^{DT}, \cdot^{LS}, \cdot^{FS})$, where N_{DT} is a set of datatypes, N_{LS} is a function that assigns lexical forms to each datatype, N_{FS} is a function that assigns a set of facet value pairs to each datatype, $\cdot^{DT}$ is an interpretation function that assigns a value space to each datatype, $\cdot^{LS}$ is an interpretation function that assigns a data value to a pair (LV, DT) where LV is a lexical form of the datatype DT, and $\cdot^{FS}$ is an interpretation function that assigns a set of data values to a pair of constraining facet and data value.

A *vocabulary* $V = (V_C, V_{OP}, V_{DP}, V_I, V_{DT}, V_{LT}, V_{FA})$ over a datatype map D is a 7-tuple, where V_C is a set of classes, V_{OP} is a set of object properties, V_{DP} is a set of data properties, V_I is a set of individuals (named and anonymous), V_{DT} is a set containing at least all datatypes of D plus the datatype $\mathsf{rdfs{:}Literal}$, V_{LT} is a set of literals $LV\hat{}\hat{}DT$ with DT a datatype and LV a lexical form for DT, and V_{FA} is the set of pairs of constraining facets and literals.

Given a datatype map D and a vocabulary V over D, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \Delta^{D}, \cdot^{C}, \cdot^{OP}, \cdot^{DP}, \cdot^{I}, \cdot^{DT}, \cdot^{LT}, \cdot^{FA})$ for D and V is a 9-tuple, where $\Delta^{\mathcal{I}}$ is a nonempty set called the object domain, Δ^{D} is a nonempty set disjoint with $\Delta^{\mathcal{I}}$ called the data domain, $\cdot^{C}$ is the class interpretation function that assigns to each class $\mathsf{C} \in V_C$ a subset C^{C} of $\Delta^{\mathcal{I}}$, similarly $\cdot^{OP}$ and $\cdot^{DP}$ interpret object and data properties as binary relations over $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and $\Delta^{\mathcal{I}} \times \Delta^{D}$ respectively, $\cdot^{I}$ assigns to each individual $\mathsf{a} \in V_I$ an element $\mathsf{a}^{I} \in \Delta^{\mathcal{I}}$, $\cdot^{DT}$ assigns to each datatype $DT \in V_{DT}$ a subset of Δ^{D}, $\cdot^{LT}$ is the literal interpretation function that assigns to a literal $LV\hat{}\hat{}DT \in V_{LT}$ a data value $(LV, DT)^{LS} \in (DT)^{DT}$, and, finally, $\cdot^{FA}$ is the facet interpretation function.

Rule Semantics and Restrictions To interpret the variables in rules, 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 if and only if every binding that satisfies the body also satisfies the head. The semantics of normal non-rule axioms and ontologies are unchanged, so an interpretation satisfies an ontology if and only if it satisfies every axiom (including the rules) in the ontology.

Given an interpretation $\mathcal{I}$ for an OWL 2 ontology $\mathcal{O}$, a *binding* $B(\mathcal{I})$ w.r.t. $\mathcal{O}$ is an interpretation that extends $\mathcal{I}$ such that $\cdot^{I}$ maps individual variables to elements of $\Delta^{\mathcal{I}}$ and $\cdot^{LT}$ maps literal variables to data values in Δ^{D}. For the DL Safe restriction we additionally require that, for each individual variable x that occurs in a DL Safe rule with $\mathsf{x}^{\mathcal{I}} = d$, there is some individual name $a \in V_I$ that occurs in $\mathcal{O}$ such that $a^{\mathcal{I}} = d$. If the ontology contains description graphs, we additionally require that the set of object property names V_{OP} can be divided into two disjoint sets of normal object properties V_{NOP} and description graph properties V_{GOP} such that no property in V_{NOP} occurs in a description graph or description graph rule and no property in V_{GOP} occurs in a normal OWL axiom or DL Safe rule. We further require that each literal variable in a data range atom also occurs in a data property atom in the body.

An atom *at* is *satisfied* by $B(\mathcal{I})$ under the conditions given in Table 2, where C is an OWL 2 class expression, D is an OWL 2 data range, OP is an OWL 2 object property expression, DP is an OWL 2 data property, x and y are individual variables or OWL individuals, and z is a literal variable or an OWL data value.

Table 2. Interpretation conditions.

at	Condition on Interpretation
ClassAtom(C x)	$\mathsf{x}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
DataRangeAtom(D z)	$\mathsf{y}^{DT} \in \mathsf{D}^{DT}$
ObjectPropertyAtom(OP x y)	$\langle \mathsf{x}^{\mathcal{I}}, \mathsf{y}^{\mathcal{I}} \rangle \in \mathsf{OP}^{\mathcal{I}}$
DataPropertyAtom(DP x z)	$\langle \mathsf{x}^{\mathcal{I}}, \mathsf{z}^{LT} \rangle \in \mathsf{DP}^{DP}$
SameIndividualAtom(x y)	$\mathsf{x}^{\mathcal{I}} = \mathsf{y}^{\mathcal{I}}$
DifferentIndividualsAtom(x y)	$\mathsf{x}^{\mathcal{I}} \neq \mathsf{y}^{\mathcal{I}}$

A binding $B(\mathcal{I})$ satisfies a body B iff B is empty or $B(\mathcal{I})$ satisfies every atom in B. A binding $B(\mathcal{I})$ satisfies a head H iff H is not empty and $B(\mathcal{I})$ satisfies every atom in H. A rule R is satisfied by an interpretation $\mathcal{I}$ iff for every binding B such that $B(\mathcal{I})$ satisfies the body, $B(\mathcal{I})$ also satisfies the head. The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology $\mathcal{O}_2$ is entailed by an ontology $\mathcal{O}_1$ iff every interpretation that satisfies $\mathcal{O}_1$ also satisfies $\mathcal{O}_2$.

Please note that although several atoms can occur in the head of a rule, these are interpreted as conjunction. In this case, the rule can equivalently be transformed into several rules with a single head atom by the standard Lloyd-Topor transformation [3]. E.g., the rule

```
Rule(Body(ObjectPropertyAtom(:singleBondsTo Variable(:x) Variable(:y)))
     Head(ObjectPropertyAtom(:bondsTo Variable(:x) Variable(:y))
          ObjectPropertyAtom(:singleBondsTo Variable(:y) Variable(:x))))
```

is equivalent to the rules

```
Rule(Body(ObjectPropertyAtom(:singleBondsTo Variable(:x) Variable(:y)))
     Head(ObjectPropertyAtom(:bondsTo Variable(:x) Variable(:y))))
Rule(Body(ObjectPropertyAtom(:singleBondsTo Variable(:x) Variable(:y)))
     Head(ObjectPropertyAtom(:singleBondsTo Variable(:y) Variable(:x))))
```

4 Concrete Syntaxes

For the sake of brevity, fully specified mappings to concrete syntaxes are not given here. They may be found in the technical report version of this paper [2]. Mappings from the syntax specified in Table 1 to RDF graphs, to OWL/XML and to the Manchester OWL Syntax have been specified. In what follows, we give a flavour of the various sytaxes using an example rule, which specifies that someone who has a sibling that is a man, has that man as a brother.

Mapping to RDF Graphs In order to maximise backwards compatibility with current syntaxes, the mapping specified for SWRL rules [7] is used.

```
<swrl:Variable rdf:about="#x"/>
<swrl:Variable rdf:about="#y"/>
<swrl:Imp>
  <swrl:body rdf:parseType="Collection">
      <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#hasSibling"/>
          <swrl:argument1 rdf:resource="#x" />
          <swrl:argument2 rdf:resource="#y" />
      </swrl:IndividualPropertyAtom>
      <swrl:ClassAtom>
          <swrl:classPredicate rdf:resource="#Man"/>
          <swrl:argument1 rdf:resource="#y"/>
      </swrl:ClassAtom>
  </swrl:body>
  <swrl:head rdf:parseType="Collection">
      <swrl:IndividualPropertyAtom>
          <swrl:propertyPredicate rdf:resource="#hasBrother"/>
          <swrl:argument1 rdf:resource="#x" />
          <swrl:argument2 rdf:resource="#y" />
       </swrl:IndividualPropertyAtom>
  </swrl:head>
</swrl:Imp>
```

OWL/XML The mapping to OWL/XML follows the same pattern as the OWL 2 specification. Names used in the extended functional syntax are mapped into elements. For example,

```
<owl:DLSafeRule>
    <owl:Body>
        <owl:ObjectPropertyAtom>
            <owl:ObjectProperty IRI="#hasSibling"/>
            <owl:IndividualVariable IRI="#x"/>
            <owl:IndividualVariable IRI="#y"/>
        </owl:ObjectPropertyAtom>
        <owl:ClassAtom>
            <owl:Class IRI="#Person"/>
            <owl:IndividuialVariable IRI="#x"/>
        </owl:ClassAtom>
    </owl:Body>
    <owl:Head>
        <owl:ObjectPropertyAtom>
            <owl:ObjectProperty IRI="#hasBrother"/>
            <owl:IndividualVariable IRI="#x"/>
            <owl:IndividualVariable IRI="#y"/>
        </owl:ObjectPropertyAtom>
    </owl:Head>
</owl:DLSafeRule>
```

Manchester OWL Syntax Rules can be represented in the Manchester OWL Syntax using a “Rule” frame. Variables are prefixed with a question marks, atoms in the rule body and rule head are separated by commas, and a dash followed by a ‘greater than’ symbol is used to separate the rule body from the head. ASCII symbols were chosen over special glyphs for conjunction and implication for ease of typing into an editor.

$$\begin{array}{l}\mathsf{Rule:}\\ \quad \mathsf{hasSibling}(?\mathsf{x}, ?\mathsf{y}), \mathsf{Man}(?\mathsf{y}) \quad -> \quad \mathsf{hasBrother}(?\mathsf{x}, ?\mathsf{y})\end{array}$$

5 Implementation

Implementation support comes in the form of an API, and reference implementation, for creating, manipulating and loading rules from various concrete serialisations, and also comes in the form of reasoning support.

API Support The OWL API has been augmented with support for working with rules. Various interfaces for representing rules, atoms, and variables have been added. Additionally, it supports parsing and rendering rules in the extended Functional Syntax, RDF based syntaxes (RDF/XML and Turtle) and OWL/XML.

Reasoning Support Pellet supports reasoning with DL Safe rules including support for most of the SWRL built-ins for numeric comparison etc. HermiT supports DL Safe rules without SWRL built-in atoms. Additionally, HermiT supports reasoning with description graphs and description graph rules.

6 Conclusion

We have presented here a syntax for rules, very similar to the SWRL syntax for rules, that is tightly integrated into the OWL 2 syntax. This syntax serves to allow rules to be part of OWL 2 ontologies and permits them to be easily presented and manipulated in OWL 2 ontology editors. We have also presented a semantics for these rules using the common DL Safe restriction and a new data range safety restriction to preserve decidability.

The DL Safe semantics for rules is not appropriate for rules used with description graphs. We have thus allowed for two types of rules in our syntax, one for use with description graphs and one for use elsewhere in the ontology, and have also presented the semantics for rules used with description graphs.

Reasoning support for these rules already exists (for example, in Pellet for DL Safe rules and in HermiT for both kinds of rules). Syntactic support for these rules is being added to the OWL API and we expect this to spur the general addition of rule support to OWL reasoners.

References

1. Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds. RIF core dialect. URL, 2009. http://www.w3.org/TR/2009/WD-rif-core-20090703/.
2. Birte Glimm, Matthew Horridge, Bijan Parsia, and Peter F. Patel-Schneider. A syntax for rules in OWL 2. Technical report, University of Oxford, 2008. http://web.comlab.ox.ac.uk/files/2445/rulesyntaxTR.pdf.
3. John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer-Verlag, 1987.
4. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, and Ulrike Sattler. Representing Structured Objects using Description Graphs. In *Proceedings of the 11th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008)*, pages 296–306. AAAI Press/The MIT Press, 2008.
5. Boris Motik, Ian Horrocks, and Ulrike Sattler. Representing Ontologies Using Description Logics, Description Graphs, and Rules. *Artificial Intelligence*, 7(2):74–89, 2009.
6. Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. OWL 2 web ontology language direct semantics. URL, 2009. http://www.w3.org/TR/2009/CR-owl2-direct-semantics-20090611/.
7. Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. SWRL: A semantic web rule language combining OWL and RuleML. URL, 2009. http://www.w3.org/Submission/SWRL/.
8. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language document overview. URL, 2009. http://www.w3.org/TR/owl2-overview/.