Interpreting SWRL Rules in RDF Graphs

Mei, J.; Boley, Harold

Publisher's version / Version de l'éditeur:

Proceedings of the International Workshop on Web languages and Formal Methods (WLFM 2005), 2006, 151, 2, 2006

NRC Publications Archive Record / Notice des Archives des publications du CNRC :
https://nrc-publications.canada.ca/eng/view/object/?id=c64e150b-a77f-4331-bd01-b497464e2e87
https://publications-cnrc.canada.ca/fra/voir/objet/?id=c64e150b-a77f-4331-bd01-b497464e2e87

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC·CNRC

Interpreting SWRL Rules in RDF Graphs *

Mei, J., and Boley, H.
2006

* Proceedings of the International Workshop on Web languages and Formal Methods (WLFM 2005). Electronic Notes in Theoretical Computer Sciencee (ENTCS). Volume 151. Issue 2. pp. 53-69. 2006. NRC 49313.

Canada

Interpreting SWRL Rules in RDF Graphs [1]

Jing Mei [2]

Department of Information Science
Peking University
Beijing 100871, China

Harold Boley [3]

Institute for Information Technology - e-Business
National Research Council of Canada
Fredericton, NB, E3B 9W4, Canada

Abstract

An unresolved issue in SWRL (the Semantic Web Rule Language) is whether the intended semantics of its RDF representation can be described as an extension of the W3C RDF semantics. In this paper we propose to make the model-theoretic semantics of SWRL compatible with RDF by interpreting SWRL rules in RDF graphs. For dealing with SWRL/RDF rules, we regard 'Implies' as an OWL class, and extract all 'Implies' rules from an RDF database that represents a SWRL knowledge base. Each 'Implies' rule is grounded through mappings built into the semantic conditions of the model theory. Based on the fixpoint semantics, a bottom-up strategy is employed to compute the least Herbrand models.

Key words: SWRL rules, RDF graphs,
model theory, fixpoint semantics, bottom-up strategy.

1 Introduction

The Semantic Web Rule Language (SWRL) is based on a combination of the Web Ontology Language [19] with the Rule Markup Language [4]. The SWRL issues dealt with in the current paper are in part responding to the

[1] The first author wishes to thank Robert Tolksdorf and his Networked Information Systems Group, Freie Universität Berlin, for providing an exciting environment for conducting this research. The second author is grateful to Mike Dean and the Joint Committee for enabling many discussions on topics related to this paper.

[2] Email: mayyam AT is.pku.edu.cn

[3] Email: Harold.Boley AT nrc.gc.ca

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

W3C team comment on the SWRL submission [16], which emphasized: "We greatly hope the path forward for SWRL includes an RDF encoding with full and correct RDF semantics, and an eye towards real user applications". Since universal variables in SWRL rules go beyond the RDF semantics [9], it has been questioned in [12] whether the intended semantics of the resultant RDF graphs can be described as a semantic extension of RDF.

In this paper we propose an RDF-compatible model-theoretic semantics of SWRL, interpreting SWRL rules in the framework of RDF graphs. This differs from the direct model-theoretic semantics of SWRL [19], whose basic idea is the definition of bindings as extensions of OWL interpretations that map variables to elements of the domain: while the mapping in [19] is s.t. a given variable appearing in different rules would assume the same value for an entire SWRL ontology, our mapping introduces rule names to permit different values for the same variable in different rules. To keep the RDF semantics of non-rule SWRL parts unchanged, we make use of an RDF resource for interpreting a variable, which will be mapped to a constant (i.e., an OWL individual or a literal value).

Moreover, since the RDF semantics permits variables as predicates, our SWRL interpretation comes in two variants, namely SWRL Full and SWRL non-Full. The former one conforms to OWL Full, while the latter one emphasizes a separation of the domain of discourse into disjoint parts (in particular, into classes, properties, individuals and variables), which makes sure that the SWRL portion, uniting OWL-DL and Datalog, preserves the standard first-order semantics. Also, a correspondence between SWRL non-Full triples and certain abstract syntax SWRL ontology "directives" (here, axioms or facts) is established. On top of that, SWRL DL-safe rules can also be achieved by restricting the universe of variables to explicitly named resources, which has been proved to be a decidable combination of OWL-DL and rules [18].

However, it should be pointed out that there is as of yet no known practical complete algorithm for reasoning in OWL-DL or OWL-Full (only OWL-Lite is more practical in this regard). Hence, our prototype system focuses on SWRL rules over RDF triples, extracting desired ground instantiations from a relational database, and computing their least Herbrand model via a bottom-up Datalog evaluation strategy. Our SWRL engine is thus available for inheritance reasoning on RDF Schema taxonomies, and we keep open an extension path towards OWL reasoning beyond RDF Schema inheritance.

Next, our motivation is presented in section 2, and the mappings to RDF graphs are elaborated in section 3. In section 4, a SWRL interpretation is introduced including the extra semantic conditions for rules, followed by the SWRL Full and non-Full interpretations plus abstract syntax correspondence. Then, SWRL DL-safe rules impose some restrictions in section 5. In section 6, we describe our implementation and apply it to 'family' rules. Related work is discussed in section 7. Finally, section 8 gives our conclusions.

2 Motivation

A rule like Def-hasUncle: "hasUncle(x,z)← hasParent(x,y), hasBrother(y,z)" is easily expressed in Datalog and corresponding decidable subsets of rule-based languages and Logic Programming (LP) [2]. However, such role chains have been proved [11] to push Description Logic (DL) into undecidable realms of First-Order Languages (FOL). For such rules it is straightforward to provide RDF graphs, e.g. Fig. 2 for the above example, but their concrete syntax can become verbose, e.g. as shown below.

s	p	o
foo:r	ruleml:body	_:b
foo:r	ruleml:head	_:h
_:b	rdf:first	_:ap
_:b	rdf:rest	_:l
_:l	rdf:first	_:ab
_:l	rdf:rest	rdf:nil
_:ap	swrl:propertyPredicate	foo:hasParent
_:ap	swrl:argument1	foo:x
_:ap	swrl:argument2	foo:y
_:ab	swrl:propertyPredicate	foo:hasBrother
_:ab	swrl:argument1	foo:y
_:ab	swrl:argument2	foo:z
_:h	rdf:first	_:au
_:h	rdf:rest	rdf:nil
_:au	swrl:propertyPredicate	foo:hasUncle
_:au	swrl:argument1	foo:x
_:au	swrl:argument2	foo:z

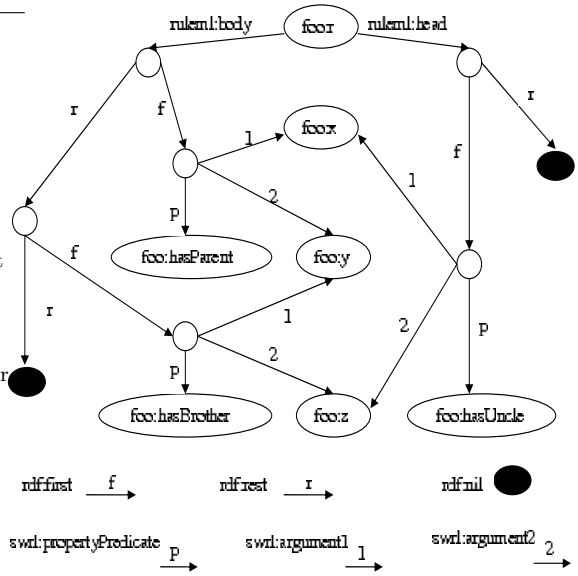

Figure 2: RDF Graph for Def-hasUncle

SWRL's RDF syntax, unlike its XML syntax, encountered the following problem: How to deal with rule variables, understood to be universally quantified outside each rule, which also means that occurrences of the same variable in different rules of the same document refer to different objects (such as when extending Fig. 2 by a 'Def-hasNiece' rule also using the variable foo:x)? [4] Indiscriminate 'grounding' of facts and rules (clauses) is no general solution either: In LP, a clause is called *ground* if it does not contain (universal) variables. By replacing every variable in a program P (namely, a finite set of clauses) with every possible value of its Herbrand universe, a grounded version of P is obtained. Generally, a clause can be viewed as a shorthand for the set of all its ground instantiations. However, in Horn logic, grounding a clause with variables can result in an infinite clause set, hence cannot be processed as a ground RDF graph. Even in Datalog, the resulting finite ground RDF graph can become too big for practical processing.

We thus focus on the interpretation of clauses with universal variables.

[4] While in RDF certain existential variables nicely correspond to blank nodes, there is no agreed-upon way to represent universal variables.

However, instead of the bindings in interpretations as defined for the SWRL semantics [12], our proposal maps a clause ID (such as "foo:r" above) together with a variable ID (such as "foo:x" above) to the variable value. This mapping can be used to ground a Datalog rule base: for each clause ID, we obtain an assignment of the clause's variables for its grounding. The resulting rules can also be required to satisfy the well-known semantic conditions of "Implies". Our approach corresponds to OWL's RDF-compatible semantics, except for the newly-introduced mapping dealing with rules.

On the other hand, using XSLT (XSL Transformations), the rule part of SWRL can be translated into LP and processed by a Datalog engine. However, as pointed out in [18], it is incorrect to compute all consequences of the OWL component first and then apply the rules to these consequences. Suppose a SWRL document consists, for example, of two rules $A(x) \leftarrow B(x), A(x) \leftarrow C(x)$, and an OWL assertion $B \sqcup C(a)$. From OWL, a is in B or C, and either way, $A(a)$ is true through the rules. But this would no longer be derived after separating OWL and rules, since neither $B(a)$ nor $C(a)$ is a consequence of OWL. Therefore, a unified interpretation of SWRL is necessary, permitting a common universe and model for both components.

Moreover, it is still a challenge to access scalable Semantic Web ontologies from applications such as in bioinformatics and e-business, since it is not clear if DL reasoners will be able to cope with realistic sets of instance data. At the other extreme, databases have been accepted as a highly effective technique to deal with mass data, and it has been pointed out in [21] that SQL provides operational constructs where *assertions* operationalize a notion of constraint rules, *views* operationalize a notion of derivation rules, and *triggers* operationalize a notion of reaction rules. In our SWRL interpreter, Sesame [5] – an open source RDF database with support for RDF Schema inferencing and querying – is used as a database of ground triples, which also act as the initial ground facts for a Herbrand model generated from Datalog rules by a fixpoint operator implemented in Java as a bottom-up engine. The derived ground facts of the model can be (filtered via constraints and) cached back into the RDF store.

3 RDF Triples from Abstract Syntax

The abstract syntax of the SWRL submission has been introduced in [12]. For example, the concrete syntax rule Def-hasUncle of section 2 becomes Abs-hasUncle in the abstract syntax:

```
Implies(foo:r
 Antecedent(foo:hasParent (I-variable(foo:x) I-variable(foo:y))
            foo:hasBrother(I-variable(foo:y) I-variable(foo:z)))
 Consequent(foo:hasUncle  (I-variable(foo:x) I-variable(foo:z))))
```

[5] http://www.openrdf.org

While the SWRL submission only gave examples, we provide here a general mapping from SWRL rules to RDF triples[6]. To extend the transformer T in Chapter 4 of the OWL semantics [19], a transformer T^* for handling rules is introduced, which will also be key to a proof that the direct and RDFS-compatible semantics are closely related (cf. section 4.3).

The unary T^*(S) returns S itself for S = individualID | dataLiteral | builtinID .

The unary T^*(S) returns URIreference for S = I-variable(URIreference) | D-variable(URIreference).

T^*(I-variable(URIreference)) $\longrightarrow$ URIreference rdf:type swrl:Variable .
T^*(D-variable(URIreference)) $\longrightarrow$ URIreference rdf:type swrl:Variable .

The binary T^*(uri, S) recursively transforms a piece of abstract rules syntax S into triples, starting at node uri (which can be a "_"-generated blank node), and returns the uri. A top-level call T^*(URIreference, S), with a fixed uri = URIreference, can be used to transform a rule S into triples rooted in URIreference. E.g., T^*(foo:r, Abs-hasUncle) returns foo:r and generates Fig. 2 in the triple store.

An empty SEQ just returns rdf:nil (possible for swrl:AtomList but not for rdf:List used in swrl:BuiltinAtom as the first occurrence).

A non-empty SEQ of $1, 2, \ldots$ elements processes its sequence right-recursively. The complete definition of the binary T^* is as follows:

T^*(uri, Implies([uri] Antecedent(antecedent) Consequent(consequent))) $\longrightarrow$
uri rdf:type ruleml:Implies .
uri ruleml:body T^*(_:b, antecedent) .
uri ruleml:head T^*(_:h, consequent) .
T^*(uri, SEQ atom_1 atom_2 ... atom_m) $\longrightarrow$
uri rdf:type swrl:AtomList .
uri rdf:first T^*(_, atom_1) .
uri rdf:rest T^*(_, SEQ atom_2 ... atom_m) .
T^*(uri, description(iobject)) $\longrightarrow$
uri rdf:type swrl:ClassAtom .
uri swrl:classPredicate T(description) .
uri swrl:argument1 T^*(iobject) .
T^*(uri, dataRange(dobject)) $\longrightarrow$
uri rdf:type swrl:DataRangeAtom .
uri swrl:dataRange T(dataRange) .
uri swrl:argument1 T^*(dobject) .
T^*(uri, individualvaluedPropertyID(iobject jobject)) $\longrightarrow$
uri rdf:type swrl:IndividualPropertyAtom .
uri swrl:propertyPredicate T(individualvaluedPropertyID) .

[6] An RDF triple (subject, predicate, object) will be written as "subject predicate object.", using spaces as separators and a period as terminator.

uri swrl:argument1 T^*(iobject) .
uri swrl:argument2 T^*(jobject) .
T^*(uri, datavaluedPropertyID(iobject dobject)) $\longrightarrow$
uri rdf:type swrl:DatavaluedPropertyAtom .
uri swrl:propertyPredicate T(datavaluedPropertyID) .
uri swrl:argument1 T^*(iobject) .
uri swrl:argument2 T^*(dobject) .
T^*(uri, sameAs(iobject jobject)) $\longrightarrow$
uri rdf:type swrl:SameIndividualAtom .
uri swrl:argument1 T^*(iobject) .
uri swrl:argument2 T^*(jobject) .
T^*(uri, differentFrom(iobject jobject)) $\longrightarrow$
uri rdf:type swrl:DifferentIndividualsAtom .
uri swrl:argument1 T^*(iobject) .
uri swrl:argument2 T^*(jobject) .
T^*(uri, builtIn(builtinID dobject$_1$ ... dobject$_l$)) $\longrightarrow$
uri rdf:type swrl:BuiltinAtom .
uri swrl:buitin builtinID .
uri swrl:arguments T^*(_, SEQ dobject$_1$ ... dobject$_l$) .
builtinID rdf:type swrl:Builtin .
T^*(uri, SEQ dobject$_1$ dobject$_2$ ... dobject$_l$) $\longrightarrow$
uri rdf:type rdf:List .
uri rdf:first T^*(dobject$_1$) .
uri rdf:rest T^*(_, SEQ dobject$_2$ ... dobject$_l$) .

A full presentation of the inductive correctness proof for the transformer T^* is beyond the scope of this paper, but we direct the interested reader to the proof elsewhere [7] .

4 RDF-Compatible Model-Theoretic Semantics

In the following, a SWRL semantics is defined, which corresponds to Chapter 5 of the OWL semantics [19] with semantic conditions added to rules.

4.1 An Extended SWRL Semantics

Definition 4.1 A SWRL interpretation, I=< $R_I, P_I, EXT_I, S_I, L_I, LV_I$ > of a vocabulary V, where V includes the RDF, RDFS and SWRL vocabularies, is an OWL interpretation of V that satisfies all of the following semantic conditions.

Firstly, Table-1 presents the SWRL universe and syntactic categories, where IRR is defined as the set of SWRL rules, IRV as variables, IRA as atoms, while IL has been defined as the set of RDF lists and IL=CEXT$_I$(S$_I$(rdf:List)).

[7] http://www.inf.fu-berlin.de/inst/ag-nbi/research/swrlengine/swrlmapping.pdf

There are SWRL built-in syntactic classes and properties:
I(ruleml:Implies), I(swrl:Variable), I(swrl:AtomList), I(swrl:Builtin),
I(swrl:Atom), I(swrl:ClassAtom), I(swrl:DataRangeAtom),
I(swrl:IndividualPropertyAtom), I(swrl:DatavaluedPropertyAtom),
I(swrl:SameIndividualAtom), I(swrl:DifferentIndividualsAtom),
I(swrl:BuiltinAtom) are all in IOC, where IOC=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{owl:Class}))$.
I(ruleml:head), I(ruleml:body), I(swrl:classPredicate), I(swrl:dataRange),
I(swrl:propertyPredicate), I(swrl:builtin) are all in IOOP,
where IOOP=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{owl:ObjectProperty}))$.
I(swrl:argument1), I(swrl:argument2), I(swrl:arguments) are all in P_I.

$\mathrm{CEXT}_I(\mathrm{S}_I(\text{ruleml:Implies}))$	= IRR	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:Variable}))$	= IRV
$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:Atom})$	= IRA	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:Builtin}))$	= IRB
$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:AtomList}))$	$\subseteq$ IL	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:IndividualPropertyAtom}))$	$\subseteq$ IRA
$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:ClassAtom})$	$\subseteq$ IRA	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:DatavaluedPropertyAtom}))$	$\subseteq$ IRA
$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:DataRangeAtom})$	$\subseteq$ IRA	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:SameIndividualAtom}))$	$\subseteq$ IRA
$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:BuiltinAtom}))$	$\subseteq$ IRA	$\mathrm{CEXT}_I(\mathrm{S}_I(\text{swrl:DifferentIndividualsAtom}))$	$\subseteq$ IRA

Table 1
SWRL universe and categories

Secondly, the characteristics of SWRL atoms and variables are presented in Table-2. With a technical trick, the well-known "syntactic sugar", i.e., (in)equality atoms of SWRL, are re-directed to the general property atoms, while specified procedures are expected for reasoning implementation. Besides, D is a datatype map [9], interpreting the built-in relations permissively, and IOT=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{owl:Thing}))$, LV_I=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{rdfs:Literal}))$, IDC=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{rdfs:Datatype}))$, IODP=$\mathrm{CEXT}_I(\mathrm{S}_I(\text{owl:DatatypeProperty}))$.

Thirdly, a new mapping is introduced, incorporating the semantic conditions for SWRL rules.

$$\mathrm{M}\text{: IRR} \times \{\text{IRV} \cup \mathrm{LV}_I \cup \text{IOT}\} \rightarrow \mathrm{LV}_I \cup \text{IOT}$$

with $\mathrm{M}(r, v)$ mapping a variable $v \in$IRV of a rule $r \in$IRR to its value $e \in \mathrm{LV}_I \cup$IOT, i.e., $\mathrm{M}(r, v) = \mathrm{e}$; or, $\mathrm{M}(r, c)$ mapping a constant $c \in \mathrm{LV}_I \cup$IOT of a rule $r \in$IRR to itself, i.e., $\mathrm{M}(r, c) = \mathrm{c}$.

If $< r, h > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{ruleml:head}))$, and $< r, b > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{ruleml:body}))$, then $r \in$IRR, h is not empty,
and h is a sequence of $h_1, ..., h_n$ over IRA,
and b is a sequence of $b_1, ..., b_m$ over IRA,
s.t. whenever b is empty or $b_j (1 \leq j \leq m)$ is true, $h_i (1 \leq i \leq n)$ is true.

Definition 4.2 An atom $a \in$IRA appearing in a rule $r \in$IRR is true iff
(1) $< a, p > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{swrl:classPredicate}))$ and
$< a, i > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{swrl:argument1}))$ and $\mathrm{M}(r, i) \in \mathrm{CEXT}_I(p)$,
(2) $< a, p > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{swrl:dataRange}))$ and
$< a, d > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{swrl:argument1}))$ and $\mathrm{M}(r, d) \in \mathrm{CEXT}_I(p)$,
(3) $< a, p > \in \mathrm{EXT}_I(\mathrm{S}_I(\text{swrl:propertyPredicate}))$ and

if	then
$< a, p >\in$EXT$_I$(S$_I$(swrl:classPredicate))	$a \in$CEXT$_I$(S$_I$(swrl:ClassAtom))$\subseteq$IRA
$< a, i >\in$EXT$_I$(S$_I$(swrl:argument1))	$p \in$IOC, $i \in$IOT$\cup$IRV
$< a, p >\in$EXT$_I$(S$_I$(swrl:dataRange))	$a \in$CEXT$_I$(S$_I$(swrl:DataRangeAtom))$\subseteq$IRA
$< a, d >\in$EXT$_I$(S$_I$(swrl:argument1))	$p \in$IDC, $d \in$LV$_I\cup$IRV
$< a, p >\in$EXT$_I$(S$_I$(swrl:propertyPredicate))	$a \in$CEXT$_I$(S$_I$(swrl:IndividualPropertyAtom))$\subseteq$IRA
$< a, u >\in$EXT$_I$(S$_I$(swrl:argument1))	$p \in$IOOP, $u \in$IOT$\cup$IRV, $v \in$IOT$\cup$IRV; OR
$< a, v >\in$EXT$_I$(S$_I$(swrl:argument2))	$a \in$CEXT$_I$(S$_I$(swrl:DatavaluedPropertyAtom))$\subseteq$IRA
	$p \in$IODP, $u \in$IOT$\cup$IRV, $v \in$LV$_I\cup$IRV
$< a, b >\in$EXT$_I$(S$_I$(swrl:builtin))	$a \in$CEXT$_I$(S$_I$(swrl:BuiltinAtom))$\subseteq$IRA
$< a, v >\in$EXT$_I$(S$_I$(swrl:arguments))	v is a sequence of $v_1, ..., v_l$ over LV$_I\cup$IRV
	$b \in$IRB, $< v_1, ..., v_l >\in$D(b)
$a \in$CEXT$_I$(S$_I$(swrl:SameIndividualAtom))	$< a,$S$_I$(owl:sameAs)$>\in$
	EXT$_I$(S$_I$(swrl:propertyPredicate))
$a \in$CEXT$_I$(S$_I$(swrl:DifferentIndividualsAtom))	$< a,$S$_I$(owl:differentFrom)$>\in$
	EXT$_I$(S$_I$(swrl:propertyPredicate))

Table 2
SWRL atoms and variables

$< a, u >\in$EXT$_I$(S$_I$(swrl:argument1)) and $< a, v >\in$EXT$_I$(S$_I$(swrl:argument2)) and $<$M(r, u), M$(r, v) >\in$EXT$_I(p)$).

Given a rule r, the maximum number of bindings is N^M where $N =$ |LV$_I\cup$IOT| is the number of constants and $M =$ |IRV| is the number of variables, hence the computational complexity of SWRL rule interpretations is $O(c \cdot N^M)$ where $c =$ |IRR| is the number of rules.

Since SWRL currently has no disjunctions or negations of atoms [17], three well-defined semantics are equivalent, that is, (1) Model Theory which is adopted by RDF semantics, (2) Proof Theory which is well-known for using SLD-resolution, (3) Fixpoint Semantics which is helpful to get the least model of a program. For SLD resolution, a series of semantic mappings serves to apply the following ground substitution, associating variables with constants, which results in grounded rules:
$\sigma = \{v \rightarrow c \mid r \in$IRR, $v \in$IRV, $c =$M$(r, v) \in$LV$_I\cup$IOT$\}$
Thus, our semantics handles a substitution (a set of variable bindings) in a disciplined manner directly in the model theory – rather than only in the proof theory – by building mappings for the grounding of rules into the semantic conditions: applied to an arbitrary rule, the mappings produce its ground instances under the current substitution. On the other hand, for its implementation, we use the fixpoint semantics for the (function-free) Horn-like top-level of rules, computing their least Herbrand model, which applies all rules in the SWRL document to all initial triple facts, to their conclusion facts, etc., until no further facts can be derived.

The following figure illustrates our method, where the resources delegating

variables are independent of those constants, and their associated relationships are established only when a rule is checked and can be reestablished for other rules.

Rules:
Def-hasUncle: hasUncle(x,z) ← hasParent(x,y), hasBrother(y,z)
Def-hasNiece: hasNiece(y,z) ← hasSibling(y,x), hasDaughter(x,z)
Facts:
<mj *hasParent* mdg> **<mdg *hasBrother* mdq>**
<mdq *hasSibling* mdg> **<mdg *hasDaughter* mj>**

Interpretation

Rule	Variable	Constant
ru=S_I(Def-hasUncle) ∈IRR	**vx=S_I(x) ∈IRV**	**oj=S_I(mj) ∈IOT**
rn=S_I(Def-hasNiece) ∈IRR	**vy=S_I(y) ∈IRV**	**og=S_I(mdg) ∈IOT**
	vz=S_I(z) ∈IRV	**oq=S_I(mdq) ∈IOT**

with mapping

M	vx	vy	vz
ru	oj	og	oq
rn	og	oq	oj

4.2 SWRL Full and non-Full Interpretation

Making the SWRL universe compatible with its RDF counterparts, we introduce an extended version called SWRL Full. In SWRL Full, as in OWL Full or RDF, an element of the universe can be an individual, a class, a property, or a variable. It is well-known that OWL Full is undecidable mainly because it allows transitive properties to occur in number restrictions, which has been strictly prohibited at the OWL-DL syntactic level. However, it has also been investigated, in [14], even $\mathcal{ALC}$ Full, the basic $\mathcal{ALC}$ Description Logic extended with metamodeling features of RDF semantics, is undecidable.

Given that our definitions are based on IOC, IOOP and P_I, SWRL Full adopts the same specifications as OWL Full.

Definition 4.3 A SWRL Full interpretation of a vocabulary V is a SWRL interpretation that satisfies: IOT=R_I, IOOP=P_I, IOC=C_I, where C_I={x∈R_I | <x,S_I(rdfs:Class)>∈EXT_I(S_I(rdf:type))}.

On the other hand, with a separation of the domain of discourse into several disjoint parts, SWRL non-Full remains standard first-order, viewed as OWL-DL plus Datalog.

Definition 4.4 A SWRL non-Full interpretation of a vocabulary V is a SWRL interpretation that satisfies: LV_I, IOT, IOC, IDC, IOOP, IODP, IOAP, IOXP, IL, IX, IRR, IRV, IRA and IRB are all pairwise disjoint.

The former ten elements are borrowed from an OWL-DL interpretation, while the latter four ones are newly-built for a SWRL interpretation.

Let us give a few examples to illustrate the distinction of SWRL"Full" and "non-Full". With SWRL Full, it is easy to describe the OWL primitive semantic conditions such that instances of OWL classes are OWL individuals.
foo:rule <foo:x rdfs:subClassOf owl:Thing>←<foo:x rdf:type owl:Class>
That is, given a rule r = S_I(foo:rule), the body is typed as swrl:ClassAtom, where swrl:classPredicate is owl:Class and swrl:argument1 is foo:x, while the head is typed as swrl:IndividualPropertyAtom, where swrl:propertyPredicate is rdfs:subClassOf, swrl:argument1 is foo:x, and swrl:argument2 is owl:Thing. Here, one argument is a variable S_I(foo:x)∈IRV whose binding is v_B=M(r, S_I(foo:x))∈IOT, while the other argument is a constant whose binding is c_B = M(r, S_I(owl:Thing)) = S_I(owl:Thing)∈IOT. Thus, for every possible binding, if v_B ∈$CEXT_I$(S_I(owl:Class))=IOC, then $\langle v_B, c_B\rangle$ ∈EXT_I(S_I(rdfs:subClassOf)), i.e., $CEXT_I(v_B)$ ⊆$CEXT_I(c_B)$=$CEXT_I$(S_I(owl:Thing))=IOT.

We observe that in SWRL non-Full, IOT and IOC are disjoint, hence it is impossible that v_B ∈IOT, v_B ∈IOC and even $CEXT_I(v_B)$ ⊆IOT, while SWRL Full permits second-order syntax allowing a variable to be bound not only to individuals but also to predicate symbols (note that IOC is the set of OWL classes, i.e., unary predicate symbols in FOL). Actually, RDF triple stores do not distinguish "Full" and "non-Full", adopting the same abstract syntax for both; however, on the semantic level the so-called "non-Full" version realizes a more restrictive style that conforms to the standard first-order semantics.

4.3 Correspondence between Abstract SWRL and SWRL non-Full

Satisfying an abstract ontology is just satisfying its directives, and satisfying the translation result of an abstract-syntax ontology to an RDF graph is just satisfying all the triples [19]. Since a SWRL ontology in the abstract syntax extends an OWL ontology with rule axioms, we present the "Implies" case here, and for the space limitation, we direct the interested reader to a complete presentation elsewhere [8].

Based on the OWL proofs [19], let V' = VO + VC + VD + VI + VOP + VDP + VAP + VXP + VR + VV + VA + VBin be a separated SWRL vocabulary, and V = VO ∪ VC ∪ VD ∪ VI ∪ VOP ∪ VDP ∪ VAP ∪ VXP ∪ VR ∪ VV ∪ VA ∪ VBin ∪ VB, where VR, VV, VA, VBin are vocabularies for SWRL rules, variables, atoms, and built-ins respectively, while the remaining vocabularies are borrowed from OWL. Let I' = <R, EC, ER, L, S, LV> be a direct interpretation of V' (cf. the SWRL submission [12]). Let I = <R_I, P_I, EXT_I, S_I, L_I, LV_I > be a SWRL non-Full interpretation of V that satisfies the triples for V', with LV_I = LV.

Proposition:
Assuming F = Implies(URIreference Antecedent(A) Consequent(C)) is a SWRL directive over V', it holds that I satisfies T^*(URIreference, F) iff I' satisfies F.

[8] http://www.inf.fu-berlin.de/inst/ag-nbi/research/swrlengine/swrlproof.pdf

Proof:
$\Leftarrow$ If I' satisfies F then, for every binding B such that B(I') satisfies the antecedent A, B(I') also satisfies the consequent C. Satisfying an antecedent A means, A is empty or B(I') satisfies every atom in A. Satisfying a consequent C means, C is not empty and B(I') satisfies every atom in C. (1) A is empty, s.t., it turns out to be an empty SEQ for T^*(_:b, A), (2) B(I') satisfies every atom in A, s.t., each transformed atom in T^*(_:b, A) is true. According to the Interpretation Conditions Table of the SWRL submission, in any case of (1) or (2), a non-empty C results in a non-empty SEQ for T^*(_:h, C), making each transformed atom in T^*(_:h, C) true, for every possible binding M($S_I(T^*$(URIreference)), $S_I(T^*(v))$)=S(v) where v is the variable URI reference occurring in F. Therefore, I satisfies T^*(URIreference, F).
$\Rightarrow$ If I satisfies T^*(URIreference, F) then, T^*(_:h, C) is not empty, i.e., C is not empty, and for every possible binding M($S_I(T^*$(URIreference)), $S_I(T^*(v))$), where v is the variable URI reference occurring in F, (1) the SEQ for T^*(_:b, A) is empty, s.t., A is empty, (2) each transformed atom in T^*(_:b, A) is true. For either case, (1) or (2), it holds that, each transformed atom in T^*(_:h, C) is true. That is, for the corresponding binding B(I') where S(v) = M($S_I(T^*$(URIreference)), $S_I(T^*(v))$), B(I') satisfies the antecedent A, B(I') also satisfies the consequent C. Thus, I' satisfies the rule F.

5 SWRL DL-Safe Rules

Early on, in the first version of SWRL [11], it has been pointed out that an OWL ontology extended with rules is undecidable, although both OWL-DL and Datalog are decidable. Recently, by restricting rules to the so-called DL-safe ones, a decidable combination of OWL-DL (i.e., the syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic) with rules is presented in [18], in addition to practical algorithms for query answering in $\mathcal{SHIQ}(\mathbf{D})$ extended with DL-safe rules.

In our case, only a trivial change to the mapping of M will make all SWRL rules DL-safe. We proceed with the introduction of DL-safe rules, followed by our safeness approach.

Definition 5.1 Let KB be a Description Logics (DL) knowledge base, where N_C is a set of DL concept names and N_R is a set of DL role names, and let N_P be a set of predicate symbols such that $N_C \cup N_R \subseteq N_P$. A DL-atom is an atom of the form $A(s)$ where $A \in N_C$, or of the form $R(s,t)$ where $R \in N_R$. A rule r is called DL-safe if each variable in r occurs in a non-DL-atom in the rule body.

Given an ordinary rule r of the form as: $A_0 \leftarrow A_1, ..., A_m$ (*)
where $A_i \in N_P$, can be made DL-safe by adding special non-DL-atoms $\mathcal{O}(x)$ to the body of rule r for any variable x occurring in r, and by adding a fact $\mathcal{O}(a)$ to the KB for each explicitly named individual a in KB. That is,

$A_0 \leftarrow A_1, ..., A_m, \mathcal{O}(x_1), ..., \mathcal{O}(x_n)$ (**)
where x_i $(1 \leq i \leq n)$ is any variable appearing in (*), in addition to the enumeration of all KB individuals as $\mathcal{O}(a)$.

In our proposal, a new set CR, being a snapshot of the closed resources from R_I, permits making all rules DL-safe, s.t. M: IRR $\times$ {IRV$\cup$LV$_I\cup$IOT} $\rightarrow$ (LV$_I\cup$IOT)$\cap$CR, where CR=$\{a \mid \mathcal{O}(a)\}$. For each variable $v \in$IRV in a rule $r \in$IRR of form (*), the conversion from (*) to (**) is realized implicitly, because M$(r, v) \in$CR and CEXT$_I$(S$_I(\mathcal{O})$)=CR, such that the atom $\mathcal{O}(v)$ is true according to the above definition, i.e., M$(r, v) \in$CEXT$_I$(S$_I(\mathcal{O})$).

However, it should be pointed out that rules in the DL community are interpreted somewhat differently. Hybrid systems integrating rules and DL usually introduce a hybrid rule being of the form $H \leftarrow B_1 \wedge ... \wedge B_m \wedge Q_1 \wedge ... \wedge Q_n$, where H, B_i are ordinary N-ary predicate symbols, and Q_j are DL queries. In a restricted version such as $\mathcal{AL}$-log [5] the DL queries act as typing constraints on the variables: (1) A backward chaining reasoner is used for the construction of a derivation of H from the B_i conjunction, while (2) leaving the Q_j conjunction to a DL reasoner for satisfiability checking. Another well-known family is CARIN [15], combining Datalog with any subset of $\mathcal{ALCNR}$: It adopts forward chaining so that the rule component is augmented by the set of DL-assertions determined by a clash-free completion of the DL component, where the DL tableaux algorithm works on such a completion to test entailment. Finally, taking negation into account, dl-programs are proposed in [6] to compute models by finite fixpoint iterations, employing a DL reasoner and an answer set engine at the same time.

6 Implementation

Our initial Java implementation [9] imports a SWRL document to Sesame (an RDF database), where all elements are decomposed into RDF triples of the form <ruleml:Implies rdf:type rdfs:Class>. As a result, a table of all 'Implies' rules is extracted from the RDF database that represents a SWRL knowledge base.

We avoid depending on an external prover and translating into it, although such a translation approach can in principle deal with SWRL rules in a first order framework, as in Hoolet [10]. For every rule, our SWRL engine implements the variable assignment of our semantics as a 1-dimensional array. The bindings represented by this vector provide ground instantiations for atoms, where each argument is indexed by '0' (for constants) or corresponding to its position in the vector (for variables). Subsequently, every 'Implies' rule should be satisfied through mappings built into the semantic conditions of the model theory; also the "sameAs" and "differentFrom" predicates in the rule body

[9] http://www.inf.fu-berlin.de/inst/ag-nbi/research/swrlengine/
[10] http://owl.man.ac.uk/hoolet/

are reduced to the (in)equality atoms.

On the other hand, to compute the least Herbrand model of all SWRL rules (which may be recursive), a bottom-up Datalog evaluation strategy is applied [10]. In this way we extend the relational database that Sesame currently deploys to a deductive database. By simulating a bottom-up SLD triple engine, it settles on a fixpoint after a finite number of iterations. However, a reasoning support combining the OWL and Datalog components in a complete manner, without resorting to an (almost) FOL prover, is an issue still being explored.

As an initial use case, a SWRL document describing certain 'family' relationships [7] was imported to Sesame, including 15 rules and 20 individuals. Having 12 assertions of "hasChild" as its primitive ones, our program derived all underlying conclusions such as 24 assertions of "hasParent", 4 of "hasUncle", 6 of "hasNiece", and so on. It was also able to infer 54 results of "hasDescendent" based on the following *recursive* rules:
hasDescendent(x,y)←hasParent(y,x)
hasDescendent(y,z)←hasParent(x,y), hasDescendent(x,z)

7 Related Work

A reduction algorithm to disjunctive Datalog programs has been developed for $\mathcal{SHIQ}(\mathbf{D})$ [14], a close relative of the $\mathcal{SHOIN}(\mathbf{D})$ description logic, of which OWL-DL is a syntactic variant. So-called DL-safe rules have been combined with disjunctive programs to increase the expressivity of the logic, without affecting decidability. On top of that, KAON2 [11] has been developed to manage OWL-DL and SWRL ontologies.

Meanwhile, since both OWL-DL and Datalog rules are decidable fragments of FOL, another optional solution is to depend on a FOL prover, translating both into a collection of formulas. Hoolet, an OWL-DL reasoner that uses the first-order prover Vampire [20], is a representative, as it has been extended to cover SWRL rules.

Alternatively, a loose coupling of the two components is attractive, i.e., running a rule engine and a DL reasoner concurrently and interchanging their conclusions by means of an interface translator, e.g. in XSLT. This entails, however, the risk of incompleteness, i.e. of not obtaining all logical conclusions. One such hybrid approach combines the Protg OWL plugin, Racer and Jess, where Racer processes OWL-DL and Jess executes production rules [7].

However, our work here intends to demonstrate that the resultant RDF graphs of SWRL can be the basis for a semantic extension of RDF. Our prototype implementation performs a mapping of RDF triples extracted from an RDF database such as Sesame. Instead of employing the customizable inferencer provided by Sesame, which processes rules defined–in their own syntax–

[11] http://kaon2.semanticweb.org/

via an external document, our engine parses SWRL rules from a SWRL/RDF document, sharing a common universe of individuals with the OWL counterparts of that document.

Our approach is amenable to adding support for OWL reasoning and flexible w.r.t. further refinements such as data filtering and updating due to Sesame's database advantages. The effect of our current implementation is similar to running a suitable translator over a SWRL document as a whole, which, however, might make OWL reasoning incomplete. As a prototype of such a translator, OWLTrans[12] provides all entailment rules of the RDF & OWL semantics, based on an encoding of RDF triples as Statement(predicate, subject, object), with Statement as the only relation; all of 'individual ID', 'class ID' and 'property ID' are encoded as constant symbols in the corresponding argument positions of Statement/3. Also, it is easy to incorporate SWRL rules in OWLTrans, where 'variable ID' becomes the variable symbol while 'classPredicate' (resp., 'propertyPredicate') is encoded as the symbol of 'class ID' (resp., 'property ID'), whose semantics is embodied in the above predefined entailment rules. The resulting rule base is quite general, including user-defined rules transformed from SWRL rules, system-defined rules employed to realize OWL entailments, and a large amount of facts dealing with the resources. Note that SWRL Full, as shown above, can only express the fragment of OWL semantics contained in Datalog, e.g. not including existential quantifiers in a rule head. FOL RuleML [3] has been employed as the target of a translator for the entire OWL semantics, where Statement/3 reduces second-order syntax to first-order syntax, making it easy to bind variables to what initially were predicate symbols.

8 Conclusions

In summary, a mapping from the abstract syntax of SWRL to an RDF concrete syntax is given, on top of which the intended semantics of the resultant RDF graphs is described as a semantic extension of RDF. Apart from specifying an RDF-compatible semantics of OWL [19], we introduce a mapping to deal with the grounding of rules, where an RDF resource corresponding to a variable in a rule will be associated with an RDF resource standing for a constant. Focusing grounded rules, we use the well-known "if-then" semantic condition, preserving the usual OWL semantics for all other directives.

For SWRL non-Full, interpretations are based on a strict separation of the domain of discourse into disjoint parts, whose correspondence to the abstract SWRL interpretation [12] has also been presented. On the other hand, SWRL Full is a more general language, providing support for metamodeling, as OWL Full does. Aiming at SWRL non-Full (i.e., OWL-DL+Datalog), SWRL DL-safe rules are considered, where, by restricting the range of our mapping, all

[12] http://www.inf.fu-berlin.de/inst/ag-nbi/research/owltrans/

SWRL rules can be made DL-safe.

On the assumption that existing RDF database systems (such as Sesame) can effectively manage SWRL knowledge by decomposing a SWRL document into an RDF graph, we implemented a prototype system to compute the least Herbrand model for SWRL rules that are in the form of RDF triples. However, the issue of a full implementation of OWL reasoning is still open (a tableaux decision procedure for $\mathcal{SHOIQ}$ – the DL underlying OWL-DL – has been recently published in [13]); similarly, as to the issue of efficient ontology-rule integrations for the Semantic Web (although there has been recent progress [1]).

References

[1] Antoniou, G. and H. Boley, "Preface: Special Issue on Rules for the Semantic Web," Elsevier Web Semantics Journal, 2005.

[2] Baral, C. and M. Gelfond, *Logic programming and knowledge representation*, Journal of Logic Programming (1994).

[3] Boley, H., M. Dean, B. Grosof, M. Sintek, B. Spencer, S. Tabet and G. Wagner, *FOL RuleML: The first-order logic web language* (2005), available at http://www.w3.org/Submission/2005/SUBM-FOL-RuleML-20050411/ and http://www.ruleml.org/fol/.

[4] Boley, H., S. Tabet and G. Wagner, *Design Rationale of RuleML: A Markup Language for Semantic Web Rules*, in: *Proc. Semantic Web Working Symposium (SWWS'01)* (2001), pp. 381–401.

[5] Donini, F. M., M. Lenzerini, D. Nardi and A. Schaerf, *AL-log: Integrating Datalog and Description Logics*, Journal of Intelligent Information Systems (JIIS) **10** (1998), pp. 227–252.

[6] Eiter, T., T. Lukasiewicz, R. Schindlauer and H. Tompits, *Combining Answer Set Programming with Description Logics for the Semantic Web*, in: *KR*, 2004, pp. 141–151.

[7] Golbreich, C., *Combining rule and ontology reasoners for the semantic web*, in: *Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings* (2004), pp. 155–169.

[8] Grosof, B. N., I. Horrocks, R. Volz and S. Decker, *Description Logic Programs: Combining Logic Programs with Description Logic*, in: *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)* (2003), pp. 48–57.

[9] Hayes, P. and B. McBride, *RDF semantics* (2004), available at http://www.w3.org/TR/rdf-mt/.

[10] Hinz, Y. K., *Datalog bottom-up is the trend*, Technical Report INSS 690, University of Maryland (2002).

[11] Horrocks, I. and P. F. Patel-Schneider, *A proposal for an OWL rules language*, in: *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)* (2004), pp. 723–731.

[12] Horrocks, I., P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, *Semantic Web Rule Language (SWRL)* (2004), available at http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[13] Horrocks, I. and U. Sattler, *A Tableaux Decision Procedure for SHOIQ*, in: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI-2005), July 31 - August 5, Edinburgh, Scotland, UK*, 2005.

[14] Hustadt, U., B. Motik and U. Sattler, *Reasoning for description logics around SHIQ in a resolution framework*, Technical Report 3-8-04/04, FZI, Karlsruhe University (2004).

[15] Levy, A. Y. and M.-C. Rousset, *CARIN: A Representation Language Combining Horn Rules and Description Logics* , in: *Proc. of the Twelfth European Conference on Artificial Intelligence (ECAI-96), Budapest, Hungary*, 1996.

[16] Marchiori, M. and S. Hawke, *Team comment on the SWRL submission* (2004), available at http://www.w3.org/Submission/2004/03/Comment.

[17] Mei, J., S. Liu, A. Yue and Z. Lin, *An extension to OWL with general rules*, in: *Third International Workshop, RuleML 2004, Hiroshima, Japan, November 8, 2004. Proceedings* (2004), pp. 155–169.

[18] Motik, B., U. Sattler and R. Studer, *Query answering for OWL-DL with rules*, in: *Proc. of the 3rd International Semantic Web Conference* (2004), pp. 549–563.

[19] Patel-Schneider, P. F., P. Hayes and I. Horrocks, *OWL web ontology language semantics and abstract syntax*, Available at http://www.w3.org/TR/owl-absyn/ (2004).

[20] Tsarkov, D., A. Riazanov, S. Bechhofer and I. Horrocks, *Using Vampire to reason with OWL*, in: *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)* (2004), pp. 471–485.

[21] Wagner, G., G. Antoniou, S. Tabet and H. Boley, *The abstract syntax of RuleML - towards a general web rule language framework*, in: *Web Intelligence*, 2004, pp. 628–631.