Reasoning Paradigms for SWRL-enabled Ontologies

Jing Mei
Department of Information Science
Peking University
Beijing 100871, China
email:mayyam@is.pku.edu.cn

Elena Paslaru Bontas
Freie Universität Berlin
Institut für Informatik
AG Netzbasierte Informationssysteme
Takustr.9, D-14195 Berlin, Germany
email:paslaru@inf.fu-berlin.de

1 Introduction

The significance of rules and rule-based representation languages is well accepted for the dissemination of the emerging Semantic Web. In this context SWRL (Semantic Web Rule Language) being a W3C proposal due to May 2004 has drawn considerable attention. With the development of SWRL Editors, such as the extension to the Protege OWL Plugin[1] which provides features for the interactive editing of SWRL rules, support for rule-based inferences on Semantic Web knowledge sources is also under development.

According to the abstract syntax of SWRL[HPSB$^+$04], an SWRL file is an OWL ontology, whose axioms are extended with rule axioms – all rules are defined as the instances of an OWL built-in class "swrl:Imp".

Technically, let S be a SWRL knowledge base, where O_C is a set of OWL class names, O_P is a set of OWL property names, and S_T is a set of OWL constants and SWRL variables. A SWRL rule has the form: $h_1 \wedge \cdots \wedge h_n \leftarrow b_1 \wedge \cdots \wedge b_m$, where $h_i, b_j, 1 \leq i \leq n, 1 \leq j \leq m$ are atoms of the form $C(i)$ with $C \in O_C, i \in S_T$, or atoms of the form $P(i,j)$ with $P \in O_P, i,j \in S_T$.

However, from the first version of SWRL, it has been pointed out, that an OWL knowledge base extended with rules is undecidable, although both components are decidable [HPS04]. Besides, as mentioned in [HPSvH03], no practical complete algorithm for reasoning in OWL DL or OWL Full has been yet proposed, while OWL-Lite is more feasible in this matter.

Consequently, in this report, we firstly present an account of the trade-offs and design decisions behind SWRL reasoners such as Hoolet[2] and KAON2[3]. Further on we propose and compare two directions and the corresponding prototypical systems coping with the mentioned problems in section 3: one is SWRL in Jess and the other is SWRL in Sesame. Jess[4] is a rule engine for the Java platform, while Sesame[5] is an open source RDF database. But it should be pointed out that, a full implementation for SWRL reasoning is still being investigated. Related work is presented in section 4, and section 5 discusses our proposal w.r.t. its limitations and future work.

2 Open Issues

The first issue to be addressed in this context is the computational complexity and the decidability. As mentioned in [HPS04], the combination of OWL and rules is undecidable. However, recently it has been investigated in [HMS04], that a restricted combination of the both would re-obtain the decidability. The second open issue is the capability of reasoning. Some approaches combine OWL and rules reasoning, while dealing with incompleteness deficiencies.

[1] http://protege.stanford.edu/plugins/owl/swrl/index.html
[2] http://owl.man.ac.uk/hoolet/
[3] http://kaon2.semanticweb.org/
[4] http://herzberg.ca.sandia.gov/jess/
[5] http://www.openrdf.org/

2.1 Computational Complexity

OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. Compared to the classical Description Logics(DL), OWL Lite is a syntactic variant of $\mathcal{SHIF}(\mathbf{D})$, while OWL DL is of $\mathcal{SHOIN}(\mathbf{D})$. Key inferences can be computed in worst case exponential time(i.e., ExpTime) in the former one and in NExpTime with the latter[HPSvH03]. Meanwhile, several DL reasoners based on tableaux algorithms, such as RACER[6] and Pellet[7], already provide effective inferences for OWL Lite, while a practical complete algorithm for OWL DL is still subject of current research. In particular, Pellet provides reasoning that is sound and complete for OWL DL without nominals ($\mathcal{SHIN}(\mathbf{D})$ in DL terminology) and OWL DL without inverse properties ($\mathcal{SHON}(\mathbf{D})$ in DL terminology). The algorithm is provably sound, but incomplete with respect to all OWL DL constructs. On the other hand, OWL Full, extending RDF and RDF(S) to a full ontology language, is an undecidable version of OWL. Even $\mathcal{ALC}$ Full, the basic $\mathcal{ALC}$ DL extended with the meta-modeling features of the RDF Semantics, is undecidable[HMS04].

Along the way, SWRL is a language combining OWL DL and OWL Lite with the unary/binary Datalog RuleML. Both components of this combination are decidable. Note that, it is the (function-free) Horn-like top-level rules that appear in SWRL, i.e., the predicate symbol of an atom in any SWRL rule can be a DL constructor, beyond the form of general rules.

Intuitively, the following simple example shows the dilemma facing to the ontology part and the rule part in a single SWRL file.
Person $\sqsubseteq \exists$ hasParent.Person
HomeWorker(x) $\leftarrow$ Person(x) $\wedge$ work(x,w) $\wedge$ live(x,w)
The first statement, a DL axiom, states that, for any person, he/she has a parent who is also a person; the second statement, a rule, states that, if the work place of a person is the same as his/her living place, then this person is a homeworker. According to the expressivity restrictions, discussed in [GHVD03], either of the two expressions fails to being asserted in its counterpart framework. This is caused by the facts that on one hand the existential quantifier cannot occur in the head of a rule, and it is impossible for DL to describe classes whose instances are related to another anonymous individual via different property paths on the other hand.

Instead of proposing an intermediate language contained within the intersection of DL and rules, SWRL insists on a unified platform to express each one in its intrinsic style. However, it cannot ensure termination of a satisfiability checking algorithm in some cases, like the above example. Asserting a single fact like *peter* is a Person, is enough to generate some undesirable results.

Figure-2.1 borrowed from [HMS04] introduces a DL blocking strategy to cope with this problem: the Person *peter* will induce the generation of Persons $x_1, x_2, \cdots$, where *peter* hasParent x_1. Being a Person, x_1 hasParent x_2 etc. Using DL algorithms, the infinite tree model is reduced to a finite one, i.e., these

[6] http://www.sts.tu-harburg.de/ r.f.moeller/racer/
[7] http://www.mindswap.org/2003/pellet/index.shtml

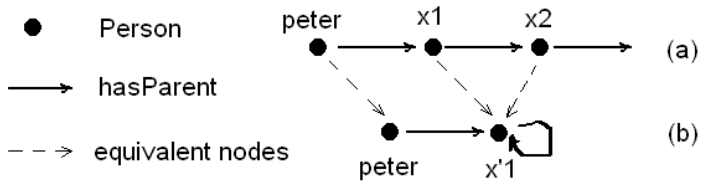

Figure 1: An Example of Decidability

infinite nodes in (a) equal to those in (b), where x_1' is the blocked node delegating all nodes of $x_1, x_2, \cdots$. Thus, DL always attempts to reach a finite tree model for evaluating its satisfiability. However, if rules are used, their substitution (i.e., the binding of a variable to constants) requires to check all instances, $x_1, x_2, \cdots$ (and x_1' is not at large), which is an infinite list. Consequently, an unrestricted combination of DL and rules easily leads to unending.

To avoid an undecidable combination, DL-safe rules have been proposed in [MSS04], requiring that each variable in a rule occurs in a non-DL-atom in the rule body. Techniques have been proposed to make rules DL-safe (see example below).

Definition: 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 set of DL individual names are constants, which together with a set of variables constitutes the set of terms T. A DL-atom is an atom of the form $A(s)$, where $A \in N_C, s \in T$, or an atom of the form $R(s,t)$, where $R \in N_R, s,t \in T$. A rule r is called DL-safe if each variable in r occurs in a non-DL-atom in the rule body.

A general rule of the form: $A_0 \leftarrow A_1, ..., A_m$ (*)
where $A_i \in N_P$, can be made DL-safe by adding special non-DL-atoms $\mathcal{O}(x)$ to the body of a 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 $\mathcal{O}(a)$.

Moreover, an algorithm for query answering in $\mathcal{SHIQ}(\mathbf{D})$ extended with DL-safe rules, employing the reduction to disjunctive datalog, has also been proposed. That is, OWL Lite plus DL-safe rules has been implemented in KAON2, whose computational complexity remains ExpTime. The case OWL DL plus rules is still ongoing work, since no decision procedure has yet been implemented for $\mathcal{SHOIN}(\mathbf{D})$.

2.2 Capability of Reasoning

Unfortunately, the DL-safe rules assure decidability at the price of inferencing power. In the following example, a rule states that, two persons who have the same parent are sibling. To make it DL-safe (i.e., to translate from * to **), the variables x, y, z are restricted by the external predicate symbol $\mathcal{O}$.
beSibling(x,y) $\leftarrow$ Person(x) $\wedge$ hasParent(x,z) $\wedge$ Person(y) $\wedge$ hasParent(y,z) (*)
beSibling'(x,y) $\leftarrow$ Person(x) $\wedge$ hasParent(x,z) $\wedge$ Person(y) $\wedge$ hasParent(y,z)
$\wedge\mathcal{O}(x) \wedge \mathcal{O}(y) \wedge \mathcal{O}(z)$ (**)
Given the facts Person(aaa), Person(bbb), hasParent(aaa,ccc), hasParent(bbb,ccc), Person(sss), Person(ttt), $\exists$hasParent.$\exists$hasParent$^-$.{sss}{ttt}, it is obvious that beSibling(aaa,bbb). However the DL-safe version (**) of beSibling does not derive beSibling(sss,ttt), because the parent of sss and ttt is not known, and the atom $\mathcal{O}(z)$ from the DL-safe rule cannot be matched to an explicitly named individual.

Hybrid approaches combining different reasoning paradigms have been proposed as a counterpart to DL-safe rules and similar solutions. However though hybrid approaches might seem attractive they risk that a rule engine and a DL reasoner run concurrently, exchanging their conclusions by means of an interface translator. For example, SWRLJessTab[Gol04] combines the Protege OWL Plugin, Racer and Jess, where Racer processes OWL DL and Jess executes production rules. However, as mentioned in [MSS04], suppose a SWRL file consists, for example, of two rules $A(x) \leftarrow B(x), A(x) \leftarrow C(x)$, and an OWL assertion $B \sqcup C(a)$. As to OWL, a is in B or C. $A(a)$ 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.

A first-order prover translating both the OWL ontology and rules into collection of axioms could be another alternative, and one representative is Hoolet using Vampire[TRBH04]. However, first order logic is undecidable and such a naive approach is highly unlikely to scale.

3 Our Proposal

Here, we proposed two options, one is SWRL in Jess, and the other is SWRL in Sesame. Although each has its benefits they do have some limitations and support for full SWRL reasoning is currently investigated.

3.1 SWRL in Jess

SWRL2Jess extends OWLTrans[8], which provides OWL2RuleML and OWL2Jess, in order to transform user-defined rules in an SWRL file to Jess rules. After the entailment rules for the OWL and RDF semantics have been predefined in a Jess rule base, the resulting SWRL2Jess rules will be appended to this file. A Jess

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

rule engine is used to handle this rule base as a whole, without distinguishing the entailment rules or the SWRL rules.

However, due to the limitations of OWLTrans, the OWL reasoning is still not fully supported, resulting in an incomplete SWRL inference service. As mentioned in the Introduction, a SWRL rule has the form:
$h_1 \wedge \cdots \wedge h_n \leftarrow b_1 \wedge \cdots \wedge b_m$
where $h_i, b_j, 1 \leq i \leq n, 1 \leq j \leq m$ are atoms of the form $C(s)$ or $P(s,t)$, C is an OWL class name, P is an OWL property name, and s,t are OWL constants or SWRL variables. According to the meaning of "Implies", if all b_j are satisfied, then all h_i should also be satisfied. Consequently, the satisfaction of every atom, occurred either in the head or in the body, depends on the matching of its instantiation, where the characteristics of its predicate symbol (i.e., unary C or binary P) has been implemented by OWL2Jess (as shown in Table 1).

atom C(x)	OWL2Jess(C)
¬D(x)	error-Msg←D(x)
A⊓B(x)	assert(A(x)∧B(x))←
A⊔B(x)	assert(A(x))←not A(x) ∧ not B(x)
{m_1,...,m_k}(x)	assert(m_1=x)←not (m_1=x) ∧ ⋯ ∧ not (m_k=x)
∀P.D(x)	assert(D(y))←P(x,y)
∃P.D(x)	assert(D(c))←P(x,y), not D(y)
∋P.{m}(x)	assert(m=y)←P(x,y), not (m=y)
∃P.D(x)	caution-Msg←not P(x,y)
∋P.{m}(x)	caution-Msg←not P(x,y)
≥n P(x)	caution-Msg ← *count*{y|P(x,y)}<n
≤n P(x)	caution-Msg ← *count*{y|P(x,y)}>n
= n P(x)	caution-Msg ← *count*{y|P(x,y)}≠ n

atom P(x,y)	OWLTrans(P)
InverseOf(Q)	assert(Q(y,x))←P(x,y)
Functional(P)	assert(y_1=y_2)←P(x,y_1) ∧ P(x,y_2)
InverseFunctional(P)	assert(x_1=x_2)←P(x_1,y) ∧ P(x_2,y)
Symmetric(P)	assert(P(y,x))←P(x,y)
Transitive(P)	assert(P(x,z))←P(x,y) ∧ P(y,z)

Table 1: SWRL Atoms

However, we do not provide the method for building a new node: suppose ∃P.D(x), for example, there is no matching of P(x,y), then such an existential semantic condition requires a new implicit node y, and P(x,y) ∧ D(y). Recalling to Figure-2.1, Person(peter) infers ∃hasParent.Person(peter), which will induce the infinite Person $x_1, x_2, \cdots$. Our incomplete inference service, being a workaround for the undecidability problem, will remind some missing information by throwing out caution-Msg, and check some inconsistent information by throwing out error-Msg.

We go back to a previous examples consisting of an OWL subclass axiom and a SWRL rule:

Person $\sqsubseteq \exists$ hasParent.Person
HomeWorker(x) $\leftarrow$ Person(x) $\wedge$ work(x,w) $\wedge$ live(x,w)
and extend it with three assertions: Person(peter), work(bob, dorm) and live(bob, dorm). Running the Jess engine would produce the caution message "no assertion of hasParent to Person peter". Following up this suggestion, we add the statement hasParent(peter, bob). Given the new statement the engine is able to infer that "bob is a Person" and "bob is a HomeWorker".

Note that, our application will enforce the assertion when it faces uncertainty. Recalling to the SWRL example consisting of $A(x) \leftarrow B(x), A(x) \leftarrow C(x)$, and $B \sqcup C(a)$, the application can not guarantee an objective result, since in this case it returns the following outputs: "now, a is a B" and "now, a is a A", despite of $B(a)$, which was decided by the engine.

3.2 SWRL in Sesame

Scalability in still an open issue for the emerging Semantic Web technologies. No solution has been proposed yet, which is able to deal with complex ontologies, especially when these ontologies contain a large set of individuals. On the other hand, in other computer science disciplines, databases have been accepted as a highly effective solution to deal with huge amounts of data. Taken this idea into account we propose an engine for SWRL rules in RDF graphs[9] on the basis of Sesame. Sesame acts as a RDF database for the initial ground triples as well as for the derived consequences.

Instead of employing the customizable inference engine provided by Sesame, which processes rules defined – in their own syntax – via an external file, our engine parses SWRL rules from a SWRL/RDF file, sharing a common universe of individuals with its OWL counterpart in the same file. A bottom-up Datalog evaluation strategy is adopted for computing the least Herbrand model of SWRL rules. As a result, the relational database that Sesame currently deploys is extended towards a deductive database.

A rule like "hasUncle(x,z)←hasParent(x,y),hasBrother(y,z)" is easily expressed in a general rule-based language such as Jess. However, for such rules it is straightforward but verbose to provide an RDF concrete syntax, as shown in Table 3.2 for the example.

Consequently, our initial Java implementation imports a SWRL file to Sesame, where all elements are decomposed into RDF triples of the form <swrl:Imp rdf:type rdfs:Class>. Furthermore, a table of all 'Imp' rules is extracted from the RDF database that represents a SWRL knowledge base, and for each rule, its type is swrl:Imp, while its body and head are both anonymous nodes. By grounding, subsequently, each 'Imp' rule will satisfy the well-known "if-then" semantic conditions, and recursive rules will also reach to the fixpoint after several loops.

Compared to the hybrid approaches mentioned in section 2.2, our approach

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

subject	predicate	object	
_:r	swrl:body	_:b	.
_:r	swrl:head	_:h	.
_:b	rdf:first	_:ap	.
_:b	rdf:rest	_:ab	.
_: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	.

Table 2: RDF Syntax for Simple Jess Rules

interprets SWRL rules in the framework of RDF graphs, which has a common universe and model for both the ontology part and the rule part. The hybrid approaches separate the two components and interpret each one independently, provided an interface for interchanging the resulting consequences. In return our prototype system is able to deal with the model of RDF Schema taxonomies as well as with user-defined SWRL rules (where RDF Schema inferencing and querying have been provided by Sesame itself). Besides it is extensible to OWL support, and more flexible w.r.t. further refinements such as data filtering and updating, due usage of the database.

3.3 A Use Case

Let us consider a SWRL family ontology. The OWL classes – Person, Man, Woman etc. – are defined as usual, and kinds of properties – hasParent, hasChild etc. – are also given in the OWL format. 15 rules, instances of the OWL built-in class "swrl:Imp", are used to declare the relationships between the above user-defined elements. Besides, 20 individuals are asserted (10 Man and 10 Woman) in this SWRL file, to represent a concrete example.

Figure-3.3 illustrate these rules, including 12 rules borrowed from a Protege ontology library[10], as well as our newly-built recursive rules "Def-hasDescendent-base", "Def-hasDescendent-iter" and an additional rule "Def-hasChild". Also, according to the Figure-3.3(cf. [Gol04]), those 20 individuals are depicted clearly.

[10]referring to http://protege.stanford.edu/plugins/owl/owl-library/family.swrl.owl

Consider that the 12 statements of "hasChild" are regarded as the only primitive assertions. In the original SWRL file, the subject of "hasChild" statements is always Man, so the following rule of "Def-hasChild" is appended, to reinforce characteristics for both Man and Woman. That is: if one Person has a child, then his/her consort also has the same child, where Person=Man⊔Woman.
hasChild(z,x)←hasChild(y,x) ∧ hasConsort(y,z)

On the other hand, the recursive "hasDescendent" procedure should also be considered. As below, the former is "Def-hasDescendent-base", for finding the direct descendent through the parent relationship; while the latter is "Def-hasDescendent-iter", for finding the indirect descendent with a recursive call to the current procedure.
hasDescendent(x,y)←hasParent(y,x)
hasDescendent(y,z)←hasParent(x,y) ∧ hasDescendent(x,z)

As a result, without more conditions other than 12 statements of "hasChild", our proposals both have drawn out the following consequences. Jess runs more quickly than our primitive reasoning engine, while Sesame provides querying visually and straightforward.

24-hasChild	14-hasSon	10-hasDaughter	08-hasNephew	06-hasNiece
24-hasParent	12-hasFather	12-hasMother	04-hasUncle	10-hasAunt
14-hasSibling	07-hasBrother	07-hasSister	54-hasDescendent	

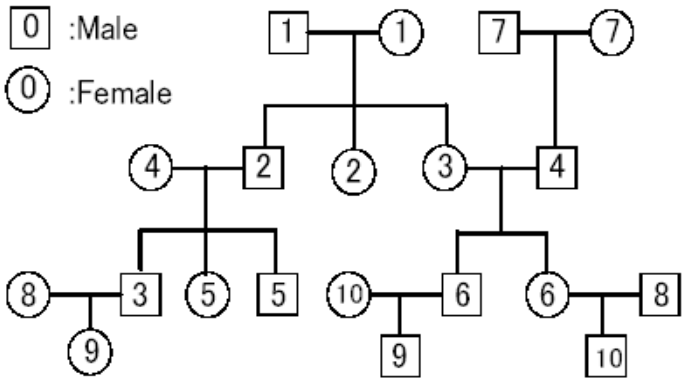

Figure 2: Individuals in a Family

4 Related Work

Related work on support for SWRL reasoning can be roughly divided into approaches reducing both components to a proper logic language, and hybrid approaches combining reasoning paradigms with interfaces for exchanging information.

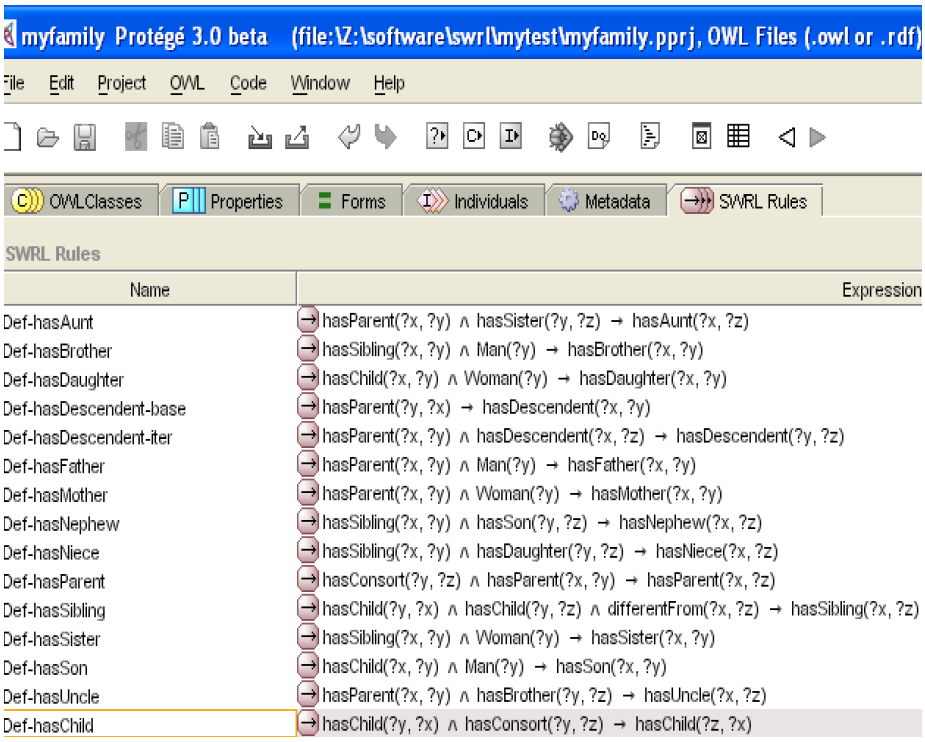

Figure 3: Rules for a Family

The first approach has different flavors: (1) first order logic is always available to be the target language for ontologies and rules – thus, as in Hoolet, a SWRL file is fed into a underlying first order prover as a collection of first order formulae. (2) algorithms discussed in [HMS04] enable to reduce certain DL knowledge base to a disjunctive program. The corresponding reasoning for $\mathcal{SHIQ}(\mathbf{D})$ plus DL-safe rules has been implemented in KAON2. (3) DLP(Description Logic Programs)[GHVD03], an intermediate KR contained within the intersection of DL and LP, can be regarded as a subset of SWRL. KAON1[11] provides support for DLP and can eventually be applied to SWRL.

The second direction has also some advantages, since each component has its own developed engines. One such hybrid approach is the development of SWRLJessTab, a Protege plugin, which enables to compute inferences with the Racer classifier and the Jess inference engine, so as to reason with rules and ontologies, both represented in OWL[Gol04]. However, its drawback is the loss of information, due to fact that the interface of the two components is unable to ensure its model completeness.

Our both approaches are scoped in the first category. For a SWRL file, a Jess rule base – consisting of: facts transformed from the SWRL ontology part, and

[11] http://kaon.semanticweb.org/alphaworld/dlp

user-defined rules transformed from the SWRL rule part, as well as predefined entailment rules serving for translating the OWL semantics – represents the knowledge base of SWRL in Jess; in the second approach an RDF database – consisting of RDF triples parsed by Sesame and inferred by our SWRL engine – represents the knowledge base of SWRL in Sesame.

5 Conclusion

SWRL in Jess reduces the OWL part to Jess facts, and appends these facts into a predefined Jess rule base, where all OWL primitive constructors are featured as entailment rules so as to implement OWL semantics. SWRL rules are transformed by a XSLT to Jess rules. SWRL reasoning is provided by running the Jess engine on the resulting Jess rule base.

SWRL in Sesame simulates a bottom-up datalog engine. With the help of Sesame, which parses a SWRL/RDF file into RDF triples stored in a relational database, this engine computes the Herbrand model of the SWRL knowledge base.

As a conclusion we are aware of the limitations of the two proposals. Comparing them we believe that SWRL in Sesame is more flexible, provided OWL reasoning in Sesame, but the optimization of our engine is planned as future work; SWRL in Jess addresses the implementation of OWL semantics directly. Ongoing work for this issue intend to eliminate the proposed workarounds by providing algorithms which try to solve the problem of undecidability in specific contexts.

References

[BG94] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 1994.

[BTW01] Harold Boley, Said Tabet, and Gerd Wagner. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In *Proc. Semantic Web Working Symposium (SWWS'01)*, pages 381–401. Stanford University, July/August 2001.

[GHVD03] 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.

[Gol04] Christine Golbreich. Combining rule and ontology reasoners for the semantic web. pages 155–169, 2004.

[GPH04] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. An evaluation of knowledge base systems for large owl datasets. In *Third Interna-*

tional Semantic Web Conference, Hiroshima, Japan, LNCS 3298, pages 274–288. Springer, 2004.

[Hin02] Yurek K. Hinz. Datalog bottom-up is the trend. Technical Report SPRING 2002, INSS 690, University of Maryland, 2002.

[HLTB04] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The instance store: Description logic reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 31–40, 2004.

[HM04] Patrick Hayes and Brian McBride. Rdf semantics. *Available at http://www.w3.org/TR/rdf-mt/*, 2004.

[HMS04] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning for description logics around shiq in a resolution framework. Technical Report 3-8-04/04, FZI, Karlsruhe University, Germany, 2004.

[HPS04] 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.

[HPSB+04] 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. *Available at http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/*, 2004.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[LPvH04] Thorsten Liebig, Holger Pfeifer, and Friedrich W. von Henke. Reasoning services for an owl authoring tool: An experience report. In *Description Logics*, 2004.

[MSS04] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. In *Proc. of the 3rd International Semantic Web Conference (ISWC 2004)*, pages 549–563. LNCS 3298, 2004.

[OWL04] OWL. Web ontology language(owl). *Available at http://www.w3.org/2004/OWL/*, 2004.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax. *Available at http://www.w3.org/TR/owl-absyn/*, 2004.

[TRBH04] Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. Using Vampire to reason with OWL. In *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, pages 471–485. Springer, LNCS 3298, 2004.