Managing Rules in Semantic Web: Redundancy Elimination and Consistency Check

Yunchuan Sun, Junsheng Zhang, Rongfang Bie, Huilin Wang


Abstract

Rules play a primary role in reasoning process to retrieve implicit and potentially useful knowledge in the Semantic Web. SWRL rules are increasingly being used to represent knowledge on the Semantic Web by using horn-like form to reason among ontologies. It is one of the key issues to keep the rule set consistent and efficient for maintaining rules in Semantic Web. In this paper, we present an approach to dealing with redundant rules among SWRL rule set. An algorithm is proposed based on the conception of literal set closure to detect and eliminate implication redundancy. Experiment results show that the proposed approach deals with the redundancy in the sample rule set efficiently and effectively. Besides, we study the consistency check of the rule set and propose algorithms to eliminate the dead-end rules, conflicts, rules with redundant antecedents, and circular rules.

Keywords: Refinement, SWRL, Redundant Rules

1. Introduction

Reasoning is always one of the key issues in the fields related to artificial intelligence. The rich semantic content facilitates the different types of reasoning in the semantic web. Semantic reasoning have been widely used in many application environments, such as web services discovery [19], answer query [20], and ect. Analogy reasoning [1] and rule based reasoning are two important and common types of reasoning. Reasoning rules play a primary role in retrieving implicit and potentially useful knowledge in the Semantic Web. Ref. [18] provides a normal form theory to regulate the semantics of semantic link network and some efficient reasoning algorithms based on reasoning rules. The Semantic Web Rule Language (SWRL) [2], based on a combination of Web Ontology Language OWL [3-4] with the Rule Markup Language (RuleML) [5-6], tends to be the rule language of the Semantic Web. SWRL uses Horn-like rules to express the reasoning relationships between terms of OWL concepts (classes, properties, individuals, literals, and etc). An SWRL rule takes the form of antecedent-consequent pairs, i.e., \( p_1 \land p_2 \land \ldots \land p_n \rightarrow q \), the antecedent is referred to the rule body while the consequent is referred to the head. The head and body consist of a conjunction of one or more atoms. An empty antecedent is treated as trivially true while an empty consequent is treated as trivially false. At present, SWRL does not support more complex logical combinations of atoms. An SWRL rule involves in OWL individuals, especially, the OWL classes and properties. Atoms in these rules take form of \( C(x) \), \( P(x,y) \), \( sameAs(x,y) \) or \( differentFrom(x,y) \), where \( C \) is an OWL description, \( P \) is an OWL property, and \( x \), \( y \) are either variables, OWL individuals or OWL data values. Given a rule,

\[
\text{Person}(?x1) \land \text{hasSibling}(?x1,?x2) \land \text{Man}(?x2) \rightarrow \text{hasBrother}(?x1,?x2)
\]

(1)

it means that a person with a male sibling has a brother requires capturing the concepts of person, male,
sibling and brother in OWL. The meaning of such a rule is obviously.

One of the key issues of maintaining the Semantic Web is to keep rule set consistent and highly-efficient in reasoning. With the extension of the Semantic Web, the size of the rule set might increase quickly. The immense expansion of the rule set due to two different cases: one is the normal expansion; the other is the more and more redundant rules. So the refinement is important for managing the rule set in the Semantic Web. Ref. [17] introduces a rule management tool to manage the SWRL rules. In the traditional artificial intelligence research, for the purpose of enhancing the efficacy and efficiency of utilizing a rule base, many approaches have been proposed for detecting and eliminating redundant and inconsistent rules in all kinds of cases [7-13]. However, almost of approaches focus on the specific environments, although some of them are useful to refine and maintain a rule set of the Semantic Web.

In this paper, we propose an approach to eliminating the redundant SWRL rules based on the conception of literal set closure. We also develop a software package to maintain and refine a rule set of SWRL. Experiment results show that the proposed approaches and the tool are efficient and effective. Besides, we study the consistency check of the rule set and propose some algorithms to eliminate the dead-end rules, confictions, rules with redundant antecedents, and circular rules.

2. SWRL rules

According to Ref. [2], a rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms. Atoms can be of the form \( C(x) \), \( P(x, y) \), \( \text{sameAs}(x, y) \) and \( \text{differentFrom}(x, y) \), where \( C \) is an OWL description or data range, \( P \) is an OWL property, \( x \) and \( y \) are either variables, OWL individuals or OWL data values, as appropriate. SWRL does not support negation atoms. Ref. [14] proposed an extension to OWL with general rules, namely ESWRL, getting classical negation and default negation involved into SWRL rules. Indeed, each atom has a negation and the meanings are shown in the Table 1.

<table>
<thead>
<tr>
<th>Negation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg C(x) )</td>
<td>( x ) is not in the class ( C )</td>
</tr>
<tr>
<td>( \neg P(x, y) )</td>
<td>( x ) is not with a value ( y ) on property ( P )</td>
</tr>
<tr>
<td>( \neg \text{sameAs}(x, y) )</td>
<td>( x ) is not same as ( y ), equivalent to ( \text{differentFrom}(x, y) )</td>
</tr>
<tr>
<td>( \neg \text{differentFrom}(x, y) )</td>
<td>( x ) is the same as ( y ), equivalent to ( \text{sameAs}(x, y) )</td>
</tr>
</tbody>
</table>

Conveniently, we call these atoms and their negations literals. Each \( p_i \) and \( q \) in (1) is all literals: atoms or atom negations. In addition, SWRL rule takes form of Horn-clause for convenience, i.e., each rule has only one literal consequent. The antecedent of each rule is a conjunction of one or more literals. Actually, a rule can be translated into one or several rule(s) with the form of Horn-clause, which is (are) equivalent to the original rule.

Each literal has three different components: negation symbol, predication name, and individual constants or variables. Negation symbol only occurs in atom negation literals. The predication name is unique for a literal. For example, Person is the predication name for \( \text{Person}(?x) \) and \( \text{Person}(?y) \). A literal might involves in 0, 1 or 2 individual constants or variables according to the atom. For example, \( \text{hasBrother}(\text{Tom}, ?x) \) involves a constant \( \text{Tom} \) and a variable \( ?x \). Two literals are matched each other if and only if:

1) both of them have the negation symbol, or neither has the negation symbol;
2) they share the identical predication names; and
3) they share the coincident constants and variables correspondingly.

As known, the order of literals in a conjunction is out of consideration. Let \( A = a_1 \land a_2 \land \ldots \land a_n \) and \( B = b_1 \land b_2 \land \ldots \land b_m \) be two conjunctions of several literals. We call that \( A \) and \( B \) are matched with each other if and only if:
1) for each $b_j$ in $B$, there is some matched literal $a_i$ in $A$; vice verse for each $a_i$ in $A$; and
2) all constant and variable individuals are coincident in $A$ and $B$ correspondingly from the global view.

For example, the rules pair
\[ \text{Person(Tom)} \land \text{hasSibling(Tom,} \, \text{?x2)} \land \text{Man(?x2)} \]
\[ \text{Person(Tom)} \land \text{hasSibling(Tom,} \, \text{?y)} \land \text{Man(?y)} \]
is a matched pair, but the pair
\[ \text{Person(Tom)} \land \text{hasSibling(Tom,} \, \text{?y)} \land \text{Man(?y)} \]
\[ \text{Person(Fred)} \land \text{hasSibling(Fred,} \, \text{?y)} \land \text{Man(?y)} \]
is not matched for the constant individual $\text{Tom}$ and $\text{Fred}$ are not coincident.

We call $B$ is matched with a fragment of $A$ if the above condition 1) changes into only for each $b_j$ in $B$, there is some matched literal $a_i$ in $A$, and the condition 2) holds.

3. Redundancy elimination

3.1. Redundant rules

Given a rule set $\mathcal{R}$ for reasoning in the Semantic Web, there might be four types of redundant rules: Equivalent rules, subsumption rules, transitive rules, and complicated rules combining the three former ones. An equivalent rule $r$ means a duplicate copy of rule $r$ in the rule set, and the difference may be the orders of the antecedent atoms or usage of different free variable(s).

**Example 3.1.** Equivalent redundant rules.

\[ \text{Person(?x1)} \land \text{hasSibling(?x1,} \, \text{?x2)} \land \text{Man(?x2)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (2a)
\[ \text{Person(?x1)} \land \text{Man(?x2)} \land \text{hasSibling(?x1,} \, \text{?x2)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (2b)

Subsumption means that two rules share the same consequent while the antecedent of one rule subsumes another’s.

**Example 3.2.** Subsumption redundant rules.

\[ \text{Person(?x1)} \land \text{hasSibling(?x1,} \, \text{?x2)} \land \text{Man(?x2)} \land \text{hasSister(?x1,} \, \text{?x3)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (3a)
\[ \text{Person(?x1)} \land \text{Man(?x2)} \land \text{hasSibling(?x1,} \, \text{?x2)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (3b)

Obviously, the antecedent of rule (3a) subsumes that of rule (3b), and their consequents are the same. So the rule (3a) is redundant in the rule set.

Transitivity occurs among three or more rules and one of them can be deduced from others.

**Example 3.3.** Transitive redundant rules.

\[ \text{Man(?x1)} \land \text{hasSibling(?x1,} \, \text{?x2)} \land \text{Man(?x2)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (4a)
\[ \text{Person(?x1)} \land \text{hasSibling(?x1,} \, \text{?x2)} \land \text{Man(?x2)} \rightarrow \text{hasBrother(?x1,} \, \text{?x2)} \] (4b)
\[ \text{Man(?x1)} \rightarrow \text{Person(?x1)} \] (4c)

We can find that the rule (4a) can be concluded from rule (4b) and (4c). So rule (4a) is a redundant one for the rule set.

Besides of the three kinds of redundancy above, some other complicated redundant rules (generally a combination of the above three kinds) might exist in a rule set.
Example 3.4. Complicated redundant rules.

\[ \text{Man}(x) \land \text{hasSibling}(x, y) \land \text{Man}(y) \land \text{hasChild}(x, z) \rightarrow \text{hasUncle}(z, y) \quad (5a) \]

\[ \text{hasChild}(x, y) \land \text{hasSibling}(x, z) \land \text{hasBrother}(y, x) \land \text{Person}(x) \rightarrow \text{hasUncle}(z, y) \quad (5b) \]

\[ \text{Person}(x) \land \text{hasSibling}(x, y) \land \text{Man}(y) \rightarrow \text{hasBrother}(x, y) \quad (5c) \]

\[ \text{Man}(x) \rightarrow \text{Person}(x) \quad (5d) \]

Rules (5b), (5c) and (5d) can derive out the rule (5a), so the rule (5a) is a redundant rule. These redundant rules cannot affect the soundness of the reasoning in general, even more transitive rules could speed up the reasoning in some cases. However, the efficiency of the reasoner performs badly if there are too many redundancies because the complexity of reasoning heavily depends on the amount of the rules. So it is important to refine the rule for the purpose of keeping the rule set in a good bulk.

If a rule \( r \) can be deduced through logical reasoning by some rules in a rule base \( R \), we say that \( r \) is implied by \( R \), denoted as \( R \models r \). If each rule of in \( R' \) is implied by \( R \), we say \( R' \) is implied by \( R \), denoted as \( R \models R' \). A rule \( r \) is trivial, if it takes the form of \( a_1 \land a_2 \land \ldots \land a_n \rightarrow a_i \), where \( a_i \) is some of \( a_1, a_2, \ldots, a_n \). It is obviously that a trivial rule always holds. Specially, a constant individual matches any free variable, but the converse is not in the case. For instance, \( \text{Man}(Tom) \) can match with \( \text{Man}(x) \), but \( \text{Man}(x) \) does not match with \( \text{Man}(Tom) \).

Definition 3.5. Let \( R \) be a rule set on the Semantic Web, and a rule \( r \in R \). If \( R \models r \), then \( r \) is an implication redundancy rule for \( R \).

From the definition, we see that \( R' \)'s function is equivalent to that of \( R-r \), for \( r \) can be replaced by some other rule(s) in \( R-r \). Thus, we should remove \( r \) from the rule set \( R \). Indeed, the above four kinds of redundant rules (i.e., equivalent, subsumption, transitive, and complicated rules) are all implication redundancy. And the complicated implication redundancy which combined by these three factors are also implication redundancy. To decide whether a rule \( r \) is redundant or not for a rule set \( R \), we should determine whether \( r \) can be derived from \( R-r \) or not.

3.2. Theory for Refinery

Definition 3.6. Let \( R \) be a rule set on Semantic Web, a new set of rules \( C(R) = \{ r | R \models r \} \) is called the closure of \( R \). For a given rule set \( R \), \( C(R) \) is just the set \( R \) unions all rules implied by \( R \). So \( C(R) \) can be regarded as a rule set with most redundant rules. Obviously, \( C(R) \) is unique for \( R \). Two rule sets \( R \) and \( R' \) are equivalent another if \( C(R) = C(R') \). In such a case, these two rule sets are with the equivalent function for the Semantic Web.

Lemma 3.7. A rule set \( R \) is equivalent to \( R' \) if and only if \( R \subseteq C(R') \) and \( R' \subseteq C(R) \).

Proof: 1) If \( R \) is equivalent to \( R' \), then \( C(R) = C(R') \). So \( R \subseteq R' \) and \( R' \subseteq R \) hold.

2) For any rule \( r \) in \( C(R') \), we can get \( R' \models r \), and \( R' \subseteq C(R) \). So \( C(R) \models r \), then \( r \in C(R') \). Of course, \( C(R') \subseteq C(R) \). Also we have \( C(R) \subseteq C(R') \). So \( C(R) = C(R') \) holds.

The primary requisition for the refinery of a rule set is to keep the equivalence of the rule set. Thus, the process of a refinery for rule set \( R \) is to get a new rule set \( R' \) by removing a subset of redundant rules from \( R \) on the condition of keeping \( R \) equivalent. Ideally, \( R' \) does not involves in any redundant rules. Such a good rule set is called a minimal cover of the original one.

Definition 3.8. The rule set \( M \) is the minimal cover of a rule set \( R \), if 1) \( C(M) = C(R) \); and 2) no rule \( r \in M \) such that \( C(M-r) = C(M) \).

For a rule set without redundant rules, the minimal cover is itself. If there are some redundancies in
a rule set, we can achieve the minimal cover by removing the redundant rules. Any rule set has a minimal cover, and a rule set could have more than one minimal cover. These different covers, if existed, are equivalent.

**Definition 3.9.** Let \( A = a_1 \land a_2 \land \ldots \land a_n \) be a conjunction of \( n \) literals and \( \mathcal{R} \) be a rule set, the closure for \( A \) with respect to \( \mathcal{R} \), denoted by \( C_{\mathcal{R}}(A) \), can be constructed as follows:

1. Let \( C_{\mathcal{R}}(A) = a_1 \land a_2 \land \ldots \land a_n \).
2. For each rule \( b_1 \land b_2 \land \ldots \land b_m \rightarrow b \) in \( \mathcal{R} \), check if \( b_1 \land b_2 \land \ldots \land b_m \) matches with a fragment of \( C_{\mathcal{R}}(A) \). If yes, let \( C_{\mathcal{R}}(A) = C_{\mathcal{R}}(A) \land b \); else goto the next rule;
3. Repeat step 2) circularly for the rule set \( \mathcal{R} \) until \( C_{\mathcal{R}}(A) \) does not change.

**Example 3.10.** \( \mathcal{R} \) is a rule set with 3 rules as following.

\[
\begin{align*}
\text{hasChild}(x_1, ?x_3) & \land \text{hasBrother}(x_1, ?x_2) \land \text{Person}(x_1) \rightarrow \text{hasUncle}(x_3, ?x_2) \quad (6a) \\
\text{Person}(x_2) & \land \text{hasSibling}(x_2, ?y) \land \text{Man}(y) \rightarrow \text{hasBrother}(x_2, ?y) \\
\text{Man}(x_1) & \rightarrow \text{Person}(x_1)
\end{align*}
\]

Let \( A = \text{Man}(x_2) \land \text{hasSibling}(x_2, ?y) \land \text{Man}(y) \land \text{hasChild}(x_2, ?z) \). We can compute \( C_{\mathcal{R}}(A) \) step by step as shown in definition (3.9). Finally, we can get that \( C_{\mathcal{R}}(A) = \text{Man}(x_2) \land \text{hasSibling}(x_2, ?y) \land \text{Man}(y) \land \text{hasChild}(x_2, ?z) \land \text{Person}(x_2) \land \text{hasBrother}(x_2, ?y) \land \text{hasUncle}(x_2, ?y) \).

In fact, the closure includes all possible consequence we can get from the initial conjunction based on the rule set. So we give an algorithm to execute reasoning in definition (3.9).

**Lemma 3.11.** \( C_{\mathcal{R}}(A) = C_{\mathcal{R} \cap \mathcal{N}}(A) \).

**Lemma 3.12.** A rule is implied by a rule set \( \mathcal{R} \) if and only if \( C_{\mathcal{R}}(A) \) includes \( b \) (the consequent of \( r \)).

From Lemma (3.12), we can get that a rule \( r : A \rightarrow b \) in a rule set \( \mathcal{R} \) is redundant for \( \mathcal{R} \) if and only if the closure \( C_{\mathcal{R}}(A) \) includes \( b \). Thus, we get an algorithm to determine whether a rule is an implication redundancy in a rule base.

### 3.3. Algorithms for Eliminating Implication Redundancy

For a rule set \( \mathcal{R} \{ r_1, r_2, \ldots, r_n \} \), the primary idea for eliminating implication redundancies is to check each rule one by one and then remove the redundant ones. In algorithm (3.13), we use a function \( \text{CLOSURE}(A, \mathcal{R}) \) to compute the closure of \( A \) with respect to \( \mathcal{R} \). Algorithm (3.14) shows the details of the function \( \text{CLOSURE}(A, \mathcal{R}) \).

In both the algorithms, we use a verb ‘match’ to determine if the given literal or conjunction matches with another conjunction. It is easy to develop such an algorithm, here we overlook the details.

**Algorithm 3.13.** Eliminating Implication Redundancy

**INPUT**: rule set \( \mathcal{R} \{ r_1, r_2, \ldots, r_n \} \), each \( r_i \) takes the form of \( A_i \rightarrow b_i \).

**OUTPUT**: rule set \( \mathcal{M} \)

\[
\begin{align*}
M &= \mathcal{R} \\
\text{FOR } i &= 1 \text{ TO } n \\
M &= M \rightarrow r_i \\
\text{IF } b_i &\text{ matches with a literal in } \text{CLOSURE}(A_i, M) \text{ THEN} \\
M &= M \rightarrow r_i \\
\text{END IF} \\
\text{NEXT } i
\end{align*}
\]

**RETURN** \( M \)

**Algorithm 3.14.** Computing the Closure

**INPUT**: rule set \( \mathcal{R} \{ r_1, r_2, \ldots, r_n \} \), each \( r_i \) with form of \( A_i \rightarrow b_i \), and a conjunction \( A = a_1 \land a_2 \land \ldots \land a_k \).

**OUTPUT**: conjunction \( B \)
\[ B = A \]
FOR \(i = 1\) TO \(n\)
IF \(A_i\) matches with a fragment of \(B\) THEN
\[ B = B \land b_i \]
END IF
NEXT \(i\)
RETURN \(B\)

### 4. Consistency check

Besides of the redundant rules, there are some rules which can lead to inconsistency during the reasoning of rule set. These rules should be eliminated to ensure the correctness of reasoning.

#### Dead-end rule

A rule is dead-end rule if its antecedent conditions are not satisfiable. The dead-end rule will never be used during the reasoning. There are two types of dead-end rules:

1) the antecedent conditions are conflict like \(p \land \neg p \rightarrow q\);
2) the antecedent conditions are never satisfiable though the antecedent conditions have no conflict.

The former type is easy to find, while the latter is hard to detect automatically, and they should be eliminated by the domain experts.

#### Confliction

Two rules are called conflict if they have the same or compatible antecedents but conflict consequents. For example, \(p \rightarrow r\) and \(p \rightarrow \neg r\) are two conflict rules. Conflict rules will lead to the inconsistent reasoning results. To eliminate the conflict rules, it is necessary to find the rule pairs whose consequent results are conflict. Then their antecedent conditions should be checked. If their antecedent conditions are the same or compatible, the two rules are conflict. The conflict rules should be eliminated by experts through editing or deleting some rules.

#### Rules with redundant antecedent

Two rules are called antecedent redundant if their consequents are identical and their antecedents are partially conflict. This type of rules should be simplified since they would slow down reasoning efficiency. For example, \(p \land q \rightarrow r\) and \(p \land \neg q \rightarrow r\) are two rules with redundant antecedent, and they can be simplified into a rule \(p \rightarrow r\).

**Algorithm 4.1. Judgment of rules with redundant antecedent**

INPUT: rule set \(R\)
OUTPUT: the rules pairs whose antecedent conditions are redundant

Steps:

1) Classify the rules in \(R\) into different groups \(R_1, R_2, \ldots, R_m\) according to the consequents;
2) For each pair of rules in a rule group \(R_i\), checking whether there is conflict antecedent. If confliction exists and the other antecedent conditions are identical, the two rules are redundant antecedents.

#### Circular rules

Circular rules consists a group of rules whose antecedents and consequents formulate a cycle. For example, \(p \rightarrow q \rightarrow w \rightarrow v \rightarrow p\) contains three circular rules: \(p \rightarrow q\), \(q \rightarrow w\), and \(w \rightarrow p\). Circular rules in a rule set could lead to the dead lock during the reasoning. It is very important to check and eliminate the circular rules for keeping the consistency of rule set.

There are two types of circular rules:

1) Simple circular rules. The antecedent of each rule is the consequent of another rule, such as \(p \rightarrow q \rightarrow w \rightarrow v \rightarrow p\) in Figure 1(a) and \(p \rightarrow q \rightarrow u \rightarrow v \rightarrow p\) in Figure 1(b);
2) Complex circular rules such as \(p \rightarrow q_1\), \(p \rightarrow q_2\), \(q_1 \land q_2 \rightarrow w\), \(w \rightarrow u\), \(u \rightarrow v\), \(v \rightarrow p\) in Figure 1(c).
Managing rules in semantic web: redundancy elimination and consistency check

Yunchuan Sun, Junsheng Zhang, Rongfang Bie, Huilin Wang


Algorithm 4.2. Check the circular rules

INPUT: rule set \( R = \{ r_1, r_2, \ldots, r_n \} \)

OUTPUT: the circular rules \( CR \)

Steps:
1) \( CR = \phi \)
2) For each \( r_i \in R \)
   2.1) Choose the consequents of rules whose antecedents are in the subset of antecedents of \( r_i \) as \( W \);
   2.2) Calculating the closures of \( W \) on rule set \( R_j = \{ r_1, r_2, \ldots, r_j \} \), denoted by \( C_j \) (\( j = 1, 2, \ldots, i-1, i+1, \ldots, n \) ), until \( C_j \) contains all the antecedents of \( r_i \) or \( j \) reaches the maximum;
   2.3) If \( C_j \) contains all the antecedents of \( r_i \), then rule set \( R' = \{ r_1, r_2, \ldots, r_j, r_i \} \) contains circular rules;
   2.4) For each rule \( r' \in R' - r_i \)
       Removing a rule \( r' \) from \( R' - r_i \), and calculate the closure \( C'_j \) of \( W \).
       If \( C'_j \) contains all the antecedent of \( r_i \), then remove the rule \( r' \) from \( R' \).
   2.5) If \( R' \not\subset \phi \) then \( CR = CR \cup R' \)
3) Return \( CR \)

5. Implementations

We have developed a software tool to eliminate the rule redundancies in a rule set according to the refinery method and algorithms mentioned above. Figure 2 is an interface of the software tool. The tool allows to import a rule set from a file or to create a new rule set by inputting new rules. We can use it to check all implication redundancies, to refine the rule set, and to execute simple reasoning tasks based on the rule set by inputting some preconditions with the reasoning module embedded in the software as shown in Figure 3.
Managing rules in semantic web: redundancy elimination and consistency check
Yunchuan Sun, Junsheng Zhang, Rongfang Bie, Huilin Wang

Figure 3. Reasoning Example Based on SWRL rule set

Table 2. The family relationship rule set

<table>
<thead>
<tr>
<th>No</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hasParent(?,?,y) ^ hasSister(?,?z) → hasAunt(?,?,z)</td>
</tr>
<tr>
<td>2</td>
<td>hasSibling(?,?,y) * Man(?) → hasBrother(?,?,y)</td>
</tr>
<tr>
<td>3</td>
<td>hasChild(?,?,y) * Woman(?) → hasDaughter(?,?,y)</td>
</tr>
<tr>
<td>4</td>
<td>hasParent(?,?,y) * Man(?) → hasFather(?,?,y)</td>
</tr>
<tr>
<td>5</td>
<td>hasSibling(?,?,y) * Woman(?) → hasMother(?,?,y)</td>
</tr>
<tr>
<td>6</td>
<td>hasSibling(?,?,y) ^ hasSon(?,?z) → hasNephew(?,?,z)</td>
</tr>
<tr>
<td>7</td>
<td>hasSibling(?,?,y) ^ hasDaughter(?,?z) → hasNiece (?,?,z)</td>
</tr>
<tr>
<td>8</td>
<td>hasParent(?,?,y) ^ hasSibling(?,?,z) → hasParent(?,?,z)</td>
</tr>
<tr>
<td>9</td>
<td>hasChild(?,?,y) ^ hasChild(?,?z) ^ differentFrom(?z, ?x) → hasConsort(?,?,z)</td>
</tr>
<tr>
<td>10</td>
<td>hasSibling(?,?,y) * Woman(?) → hasSister (?,?,y)</td>
</tr>
<tr>
<td>11</td>
<td>hasChild(?,?,y) * Man(?) → hasSon(?,?,y)</td>
</tr>
<tr>
<td>12</td>
<td>hasParent(?,?,y) ^ hasBrother(?,?z) → hasUncle(?,?,z)</td>
</tr>
<tr>
<td>13</td>
<td>hasChild(?,?,y) → hasParent(?,?,y)</td>
</tr>
<tr>
<td>14</td>
<td>hasParent(?,?,y) → hasChild(?,?,y)</td>
</tr>
<tr>
<td>15</td>
<td>hasConsort(?,?,y) ^ hasChild(?,?z) → hasChild(?z, ?y)</td>
</tr>
<tr>
<td>16</td>
<td>hasDaughter(?,?,y) → hasChild(?,?,y)</td>
</tr>
<tr>
<td>17</td>
<td>hasSon(?,?,y) → hasChild(?,?,y)</td>
</tr>
<tr>
<td>18</td>
<td>hasBrother(?,?,y) → hasSibling(?,?y)</td>
</tr>
<tr>
<td>19</td>
<td>hasSister(?,?,y) → hasSibling(?,?,y)</td>
</tr>
<tr>
<td>20</td>
<td>hasFather(?,?,y) → hasParent(?,?,y)</td>
</tr>
<tr>
<td>21</td>
<td>hasMother(?,?,y) → hasParent(?,?,y)</td>
</tr>
<tr>
<td>22</td>
<td>hasParent(?,?,y) * Woman(?) → hasDaughter(?,?,y)</td>
</tr>
<tr>
<td>23</td>
<td>hasParent(?,?,y) * Man(?) → hasSon(?,?,y)</td>
</tr>
<tr>
<td>24</td>
<td>hasSon(?,?,y) ^ hasSon(?,?z) ^ differentFrom(?z, ?x) → hasConsort(?,?,z)</td>
</tr>
<tr>
<td>25</td>
<td>hasDaughter(?,?,y) * hasDaughter(?,?,z) → hasConsort(?,?,z)</td>
</tr>
<tr>
<td>26</td>
<td>^ differentFrom(?,?z) → hasConsort(?,?,z)</td>
</tr>
<tr>
<td>27</td>
<td>hasSibling(?,?,y) * Woman(?) * Man(?) → hasBrother(?,?,y)</td>
</tr>
</tbody>
</table>
We have done some experiments with the following rule set based on the family example in [16] as shown in Table 2. The tool finds out all implication 5 redundant rules in the sample rule set. The result shows the possibility and effect of our method and algorithms. Furthermore, some experiments result on reasoning tasks are listed in the Table 3, which show the soundness of the reasoning algorithm.

<table>
<thead>
<tr>
<th>Preconditions</th>
<th>Reasoning Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasSon(?x,?y)^hasSon(?z,?y)^differentFrom(?z, ?x)</td>
<td>hasConsort(?x,?y)^hasChild(?z,?y)^hasParent(?y, ?x)</td>
</tr>
<tr>
<td>hasSibling(Bob, ?y)^Woman(?y)</td>
<td>hasSister(Bob, ?y)^hasSibling(Bob, ?y)</td>
</tr>
<tr>
<td>hasParent(Tom, Mary)^hasSister(Mary, Lily)</td>
<td>hasChild(Mary,Tom)^hasAunt(Tom,Lily)^hasNephew(Lily,Tom)</td>
</tr>
</tbody>
</table>

6. Conclusions

This paper investigates the approaches to refining SWRL rule set. Reasoning rules play a primary role in the process of reasoning to retrieve implicit and potentially useful knowledge in the Semantic Web. Based on OWL and RuleML, the SWRL would be the Semantic Web rule language. In maintaining Semantic Web, one of the key issues is maintaining rule set to keep the consistency and highly efficiency. With the extension of the Semantic Web, the size of rule set might become larger and larger. So one of the primary issues to manage and maintain the Semantic Web is how to process the redundant rules and inconsistent rules in the SWRL rule set.

In this paper, we propose an approach to dealing with the redundant rules in the rule set of Semantic Web represented by SWRL. We also develop a software tool to maintain and to refine a rule set on the Semantic Web with SWRL. The presented algorithm for detecting and eliminating implication redundancy is based on the closure of literal set. The experiment shows the approaches and tool are efficient and effective. The software also provides a module to implement the reasoning based on the rule set with SWRL. Besides, we also study the consistency check of the rule set and propose some algorithms to eliminate the dead-end rules, conlfrictions, rules with redundant antecedents, and circular rules.

7. Acknowledgment

This research has been partially supported by ISTIC research foundation projects YY-2010018, ZD2010-3-3, XK2010-6 and Beijing Excellent Talent Fostering Foundation 2010D00901200001.

8. References.


