Abstract

In this paper general mechanisms and syntactic restrictions are explored in order to specify and merge rule bases in the Semantic Web. Rule bases are expressed by extended logic programs having two forms of negation, namely strong (or explicit) and weak (also known as default negation or “negation as failure”). The proposed mechanisms are defined by very simple modular program transformations, and integrate both open and closed world reasoning. These program transformations are shown to be appropriate for the two major semantics for extended logic programs: answer set semantics and well-founded semantics with explicit negation. Moreover, the results obtained by both semantics are compared.

1 Introduction

The Semantic Web [3] aims at defining formal languages, and corresponding tools, enabling automated processing and reasoning over (meta-)data available from the Web. Logic and knowledge representation play a central role, but the distributed and worldwide nature of the Web bring new interesting research problems. In particular, the widely recognized need of having rules in the Semantic Web [10, 18] has restarted the discussion of the fundamentals of closed-world reasoning and the appropriate mechanisms to implement it in rule systems, such as the computational concept of negation-as-failure.

The classification if a predicate is completely represented or not is up to the owner of the knowledge base: the owner must know for which predicates there is complete information and for which there is not. Unfortunately, neither classical logic nor standard Prolog supports the distinction between “closed” and “open” predicates. Classical logic supports only open-world reasoning. On the contrary, most Prolog systems support only closed-world reasoning, as negation-as-failure is the only negation mechanism supported (notable exceptions are XSB [19, 16] and CIAO [13, 12]). In this paper, we resort to two major semantics of extended logic programs which allow the distinction between open and closed predicates, and illustrate their application to the declaration and construction of rule bases in the Semantic Web.

The paper is organized as follows. In the next section, the use of extended logic programming is explored to represent open and closed world reasoning, providing general mechanisms for achieving this. Section 3 defines new language mechanisms for sharing and integrating knowledge in the Semantic Web. The paper finishes with comparisons and conclusions.

---

1This research has been partially funded by European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REVERSE number 506779 (cf. http://reverse.net).
2 Open and Closed World Assumption

Rule bases are sets of extended logic programming rules of the form

\[ L_0 \leftarrow L_1, \ldots, L_m, \sim L_{m+1}, \ldots \sim L_n \]  

(1)

where each \( L_i \) (with \( 0 \leq i \leq n \)) is an objective literal, i.e. either an atom \( A(t) \) or the strong negation of an atom \( \neg A(t) \), where \( t \) is a sequence of terms. Variables are prefixed with a question mark symbol (\(?\)) for predicates, constants and function symbols can start with small and capital letters. It is assumed that a fixed first order logic alphabet is given, and only extended Herbrand interpretations are considered (sets of objective literals). In particular, a non-ground rule in an extended logic program stands for the set of ground rules obtained by instantiating logical variables with elements from the Herbrand domain. Notice that implicitly we are using a domain closure assumption which might not be acceptable in some situations. Without loss of generality, only ground programs are considered in the subsequent theoretical results. Furthermore, we restrict the discussion to DATALOG programs over a finite number of constants in order to guarantee decidability of reasoning. We define by \( C_{SEM}(P) \) the set of objective literals which can be concluded from the extended logic program \( P \) under semantics SEM. Here we consider only skeptical answer set semantics [7], denoted by subscript \( SEM = AS \), and well-founded semantics with explicit negation [15, 1], denoted by subscript \( SEM = WFSX \). The reader is referred to the literature for details.

The arbitrary uncontrolled use of weak negation in the Semantic Web is regarded problematic and unsafe. However, local closed world assumptions and scoped negation as failure have been identified as desirable and necessary for the Semantic Web [8, 11, 17, 24, 2]. The difficulty lies on the definition of simple mechanisms that can be easily explained to ordinary users, and have nice mathematical properties. For this reason, we propose a classification of predicates which cover the whole gamut of alternatives.

Suppose that a predicate \( A \) is being defined in the Semantic Web. Besides normal (or ordinary) predicates, we define the classes of objective, open and closed predicates, which are summarized in Figure 1. The top-half boxes contain the user’s predicate definitions and are always sets of objective rules, i.e. rules which do not contain weak negation but might contain strongly negated literals, in particular the head of rules might be \( A(t) \) or \( \neg A(t) \). The bottom-half boxes contain special rules, added by the system, which characterize each type of predicate.

Objective predicates are defined by objective rules which do not contain weak negation at all. Since strong negation is monotonic, these predicates can be freely used in the Semantic Web without any restriction. These predicates are partial since it may be the case that neither \( A(\overline{t}) \) nor \( \neg A(\overline{t}) \) hold in a model, where \( \overline{t} \) is a sequence of constants. On the other hand, open predicates have the following two additional rules, denoted \( openRules(A) \):

\[ A(\overline{t}) \leftarrow \neg A(\overline{t}) \quad \neg A(\overline{t}) \leftarrow A(\overline{t}) \]

In answer set semantics, these specify that either \( A(\overline{t}) \) is true or \( \neg A(\overline{t}) \) is true in each model (answer set), thus forcing totalness.

Finally, closed predicates are complemented by one and only one of the previous two rules, called default closure rules, and denoted by \( negClosure(A) \) and \( posClosure(A) \), respectively. This provides a mechanism for making closed world assumptions: either by making true what is not concluded false or by making false what is not concluded true.

Notice that in the move from all predicates being objective to some being open and then closed, new conclusions might be obtained, as the following major Theorem shows:

**Theorem 1** Let \( A \) be an objective predicate in extended logic program \( P \) where all predicates are either objective or open. Then,

- \( C_{SEM}(P) \subseteq C_{SEM}(P \cup openRules(A)) \)
- \( C_{SEM}(P \cup openRules(A)) \subseteq C_{SEM}(P \cup posClosure(A)) \)
Objective predicate | Open Predicate | Closed Predicate
--- | --- | ---
$L_0 \leftarrow L_1, \ldots, L_m$

\( A(\bar{x}) \leftarrow \sim A(\bar{x}) \)

\( \neg A(\bar{x}) \leftarrow \sim A(\bar{x}) \)

\( \neg A(\bar{x}) \leftarrow \sim A(\bar{x}) \) or \( \neg A(\bar{x}) \leftarrow A(\bar{x}) \)

Figure 1: Declarations for a predicate \( A \) (the predicate of \( L_0 \) is \( A \))

- \( C_{SEM} (P \cup openRules(A)) \subseteq C_{SEM} (P \cup negClosure(A)) \)

with \( SEM = AS \) or \( SEM = WFSX \).

For the case of \( WFSX \) semantics the first containment is in fact an equality, i.e. \( C_{WFSX} (P) = C_{WFSX} (P \cup openRules(A)) \).

The previous theorem cannot be generalized when some predicate is closed in \( P \). This is expected due to the non-monotonic nature of weak negation under both \( AS \) semantics and \( WFSX \) semantics.

\( WFSX \) is a tractable semantics which approximates Answer Sets, and therefore is a good candidate for defining the semantics of rule bases in the Semantic Web. Also, under \( WFSX \) no new objective conclusions are obtained by declaring predicates open. This is expected since entailment in \( WFSX \) can be computed in polynomial time, while entailment in \( AS \) is coNP-complete. This is the tradeoff between expressivity and complexity of reasoning. However, \( WFSX \) and \( AS \) are not unrelated:

**Theorem 2** [15] Let \( P \) be an extended logic program, then \( C_{WFSX} (P) \subseteq C_{AS} (P) \).

However, the existence of an undefined truth-value in \( WFSX \) might affect the intuition in some particular cases, namely for closed predicates: this is the price to pay for guaranteeing tractability of reasoning. Aside that, both semantics assure the monotonicity of reasoning in the presence of only objective and open predicates:

**Theorem 3** Let \( P \) and \( Q \) be two extended logic programs where all predicates are either objective or open. Then,

- \( C_{AS} (P) \subseteq C_{AS} (P \cup Q) \)
- \( C_{WFSX} (P) \subseteq C_{WFSX} (P \cup Q) \)

However, the previous result does not hold whenever closed predicates are included in \( P \) or \( Q \). The above theorems are explored in the next section for defining modular programming techniques to be used in the Semantic Web.

3 Modularity in the Semantic Web

In this section we study the mechanisms in order to be able to express the necessary context to use strong and weak negations safely in the Semantic Web environment. The discussion is abstract and independent of any rule engine. Currently, there is no notion of scope or context in the Semantic Web: all knowledge is global and all kinds of unexpected interactions can occur. The success of the Semantic Web is impossible without any form of modularity, encapsulation, information hiding and access control. The issue of modularity in logic programming has been actively investigated during the 90s, for a survey see [4]. Here we follow a typical approach similar to the import/export mechanisms of Prolog, but we will be concerned with the combination
of open and closed world reasoning and other particularities of the Semantic Web. In particular, the following four levels of context and their interaction must be taken into account:

- The Semantic Web context;
- The application context, corresponding to the context where a user or Semantic Web agent loads, asserts or consumes the knowledge provided by rule bases in the Semantic Web;
- The rule base context, where the Semantic Web developer encapsulates a set of related rules and facts (predicates);
- The predicate context, which can be either global or local;

Rule bases are made available in the Semantic Web, and users or applications load or assert them explicitly into their application contexts. The connection to an external knowledge base should always be equivalent to loading it locally, but without the need to explicitly do that. When a user or application loads or asserts knowledge, it may express that nonmonotonic reasoning forms may be rejected or allowed, or can force the deduction mechanisms to use only rules which extract safe knowledge in the Semantic Web context. The knowledge base programmer may use nonmonotonic constructs, knowing that these constructs might be inhibited or forbidden. The producer of knowledge might also express that the predicates he/she is declaring cannot be defined elsewhere, and may declare hidden predicates which are not visible in the Semantic Web. Furthermore, a knowledge base might use all the available knowledge in the application context, or get it explicitly from particularly loaded rule bases. It is mandatory that, by default, all knowledge inferred from the combination of several knowledge bases and asserted facts into the application context must be safe in the Semantic Web.

The challenge is to provide simple mechanisms in order to guarantee the fulfilment of the previous requirements. Obviously, the syntax of extended logic programs should be augmented with declarations to state the visibility of a predicate, its context, and whether it is normal (i.e. unrestricted), objective, open or closed. It is also necessary to express how external information to the knowledge base is incorporated into it. These can be attained with the declarations **defines** and **uses** with the syntax in BNF notation presented in Figure 2. The **defines** declaration specifies which predicates are defined (and exported) in the knowledge base, their scope and visibility, as well as type. The **uses** declaration describes which predicates are used (imported) from other rule bases or from the Semantic Web, and might change the original type of the predicate. Notice that predicates and rule bases are all identified by absolute IRIs (Internationalized Resource Identifiers).

The scope plays a fundamental part, and describes what is the context of the predicate(s) and may take one of the following values, with the following corresponding limitations and meaning:

"**global**": a predicate declared global is visible outside the knowledge base, and intends to capture predicates being defined in the Semantic Web. Moreover, the predicate can be defined elsewhere in other rule bases but it must be either objective or open\(^2\). Additionally, it can be optionally declared which rule bases can use the predicate; if omitted, it can be used everywhere. All predicates in the RDF and RDFS vocabularies are global and open. This is the default scope.

"**local**": a local predicate can be used outside the rule base where it has been defined, but cannot be defined by any other knowledge base in the Semantic Web. A local predicate can be of any type (objective, open, closed and normal) and, as before, the user can state the rule bases where it can be used. The rule base defines the scope for a closed predicate, and the closure rule

\(^2\)For simplicity, this constraint is not enforced in the grammar.
Figure 2: defines and uses declarations

may be inhibited by the consumer of the knowledge in the uses statement. If the predicate is normal, any form of negation can be used in its definition, and its use can be forbidden by the consumer of the knowledge, again with the uses statement.

"internal": predicate is internal to the rule base and cannot be used outside the rule base. Again, the rule base defines the scope for the evaluation of weak negation.

Additionally, it is required that objective, open and closed predicates cannot use (directly or indirectly) normal predicates on their definitions. This prevents unintended use of weak negation in the Semantic Web.

The visibility provides a basic security mechanism, but trust and authorization could be much improved, for instance using the PeerTrust language [6]. These issues are orthogonal to present proposal but can be easily integrated due to the logical nature of our work.

The uses declaration specifies the rule bases providing the definitions of global and local predicates that can be used by the importing rule base. Notice that the importer can specify what types of predicates (reasoning) he/she is willing to accept. The exporter must provide the answers according to the cases specified in Table 1. For instance, suppose that a rule base $< RB_A >$ defines a closed predicate $P$ with

Table 1: Combination of reasoning modes

<table>
<thead>
<tr>
<th>uses (consumer)</th>
<th>objective</th>
<th>open</th>
<th>closed</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed</td>
<td>objective</td>
<td>open</td>
<td>closed</td>
<td>error</td>
</tr>
<tr>
<td>open</td>
<td>objective</td>
<td>objective</td>
<td>objective</td>
<td>error</td>
</tr>
</tbody>
</table>

$< RB_A >$ defines local closed $P$.

However, the uses statement in rule base $< RB_B >$ declares that it is only willing to accept the conclusions obtained by opening the predicate $P$ in $< RB_A >$:

$< RB_B >$ uses open $P$ from $< RB_A >$.

Rule base $< RB_A >$ should only provide answers to queries of $P$ from $< RB_B >$ as if all closed predicates in $< RB_A >$ were open. If $< RB_B >$ uses predicate $P$ of $< RB_A >$ in objective mode, then all predicates in rule base $< RB_A >$ are considered objective when computing the queries to $P$ from $< RB_B >$.

In other words, the reasoning mode should also be propagated to the predicates used in $< RB_A >$, whenever these predicates are necessary to answer the original query. Finally, we would like to note that there are subtle issues involved in the above mechanisms, namely the possibility of mutual dependencies between rule bases, which should be addressed in implementations. A runtime error is thrown when the exporter declares a local predicate normal but the importer uses one
of the limited predicate reasoning forms: objective, open or closed. This behaviour corresponds to rejecting by the importer the uncontrolled use of weak negation in the Semantic Web. Finally, by default, all predicates in uses are in open mode. According to the results of the previous section, the default declarations guarantee that reasoning is monotonic.

A knowledge base might define and use the same predicate, but not all combinations are possible. The various allowed combinations are presented in Table 2. Obviously, it is an error to globally or locally define a used local predicate; this goes against the notion that there is a sole provider for a local predicate. However, it is allowed to internally redefine a local predicate of a different rule base, since it is not made public. In particular, one might close an open local predicate of a different provider since this is only for internal use.

### Table 2: Defining and using the same predicate

<table>
<thead>
<tr>
<th></th>
<th>defines</th>
<th>uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>global</td>
<td>allowed</td>
<td>error</td>
</tr>
<tr>
<td>local</td>
<td>error</td>
<td>error</td>
</tr>
<tr>
<td>internal</td>
<td>allowed</td>
<td>allowed</td>
</tr>
<tr>
<td>global</td>
<td>local</td>
<td>internal</td>
</tr>
</tbody>
</table>

4 Comparison and Conclusions

The notion of localized closed world assumptions has been proposed for instance in [8]. The idea is to have syntactic mechanisms in the Semantic Web languages (like DAML+OIL or OWL) to express that a predicate is closed, i.e. something which cannot be inferred can be assumed false: this is a usual assumption in logic programming (negation as failure, by default or weak) and relational databases (the set difference operation of relational algebra). The major problem with the proposal of Hefflin and Munoz-Avila is the use of a Clark’s completion like approach, which is well-known to suffer from serious problems when applied to knowledge based systems [22, 21], even without negation.

The notion of scoped negation as failure has also been suggested by several authors, see for instance [11, 17], and systems like FLORA-2 [25] do support it. Both FLORA-2 and TRIPLE [23] support modularity constructions, which are essential for deployment of inference engines in the Semantic Web. However, in contradistinction to the existing systems, we define the notion of objective, open and closed predicates, their semantically compatible definition, as well as languages constructs for controlling knowledge in the Semantic Web. The combination of open world and closed world reasoning in the same framework is also proposed in [2], where the ERDF stable model semantics of Extended RDF knowledge bases is developed, based on partial logic [9]. However, modularity issues are not considere there. The existence and combination of all our proposed mechanisms in a single language is a novelty, to the best of our knowledge.

The language is intuitive to use and gives absolute freedom to producers and consumers of knowledge in the Semantic Web. It can be implemented with the existing technology, and can support and integrate different inference engines ranging from relational databases to state-of-the-art inference engines, including description logic reasoners. Both tractable and more complex forms of inference are also easily syntactically identified and delimited.

The semantics of the constructs can be defined via immediate program transformations, for which the rationale and corner-stone elements have been introduced in this paper. The full semantics of the language will be presented in a subsequent paper.

It should be noted that our method of using a default closure rule for closing a predicate does not work in the presence of disjunctive information: whenever there is any kind of disjunctive information about predi-
cate \( p \), then \( \sim p(\bar{c}) \) (resp. \( \sim \neg p(\bar{c}) \)) cannot be derived due to the presence of intended models containing \( p(\bar{c}) \) (resp. \( \sim \neg p(\bar{c}) \)). Therefore, in addition to adding a default closure rule for \( p \), any disjunctive information about \( p \) should be disallowed. This is an issue of preventing knowledge base users (and other information sources) to input any form of disjunctive information about the extension of \( p \). Such a restriction seems to be justified by the fact that closing a predicate means the knowledge-based system has complete knowledge and control over its extension.

Another issue is that the method of using default closure rules only works for partial predicates, because in the case where \( p \) is total, \( p(\bar{c}) \) cannot fail due to the fact that there is a state of affairs (a minimal or stable model) in which \( p(\bar{c}) \) holds. Since one may argue that it makes more sense to close total predicates, we just point out that this may be viewed as a problem, but we do not propose a solution to it.

There are still some important practical problems to be addressed at the implementation level for which solutions exist, but for lack of space cannot be presented in this work. Furthermore, the issue of contradiction is not addressed here, but the results of Section 2 can be adapted for existing paraconsistent semantics for extended logic programs, namely [14, 1, 5, 20]. A prototypical implementation is underway, using immediate extensions to RuleML markup language [18].

5 Acknowledgments

This work has benefited from discussions with Luís Moniz Pereira and José Júlio Alferes.

References


