Towards an Abstract Syntax and Direct-Model Theoretic Semantics for RuleML

Adrian Giurca and Gerd Wagner

Institute of Informatics, Brandenburg University of Technology at Cottbus
{Giurca,G.Wagner}@tu-cottbus.de

Abstract. This paper contains a proposal of an abstract syntax and a model theoretic semantics for NafNegDatalog, sublanguage of RuleML [9]. The model-theoretic semantics use the partial logic ([7], [10]) to provide an interpretation and a satisfaction relation, and provide a formal meaning for RuleML knowledge bases written in the abstract syntax.

Keywords: rule markup languages, RuleML, abstract syntax, semantics, partial logic.

1 Introduction

The RuleML Initiative [9] started in August 2000 during the Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000). It has brought together expert teams from several countries, including leaders in Knowledge Representation and Markup Languages, from both academia and industry.

The RuleML Initiative is developing an open, vendor neutral XML/RDF-based rule language. This will allow for the exchange of rules between various systems including distributed software components on the Web, heterogeneous client-server systems found within large corporations, etc. The RuleML language offers XML syntax for rules Knowledge Representation, interoperable among major commercial and non-commercial rules systems.

During this period 12 sublanguages was developed (see [2], Figure 2). Each sublanguage has a proposed XML syntax and DTD for validation and from version 0.85 we have also an XML Schema for validation.

In this paper we propose an abstract syntax and a model theoretic semantics for the NafNeg-Ur-Datalog sublanguage of RuleML.

2 The Abstract Syntax

The abstract syntax is specified here by means of a version of Extended BNF, very similar to the EBNF (based on Wirth's definition) notation used for XML. Nonterminals are enclosed between angle brackets (< and >) and terminals are in boldface. Alternatives are either separated by vertical bars (|) or are given in different productions. Components that can occur at most once are enclosed in square brackets ([...]); components that can occur any number of times (including zero) are enclosed in braces ({...}). Whitespace is ignored in all the productions.

Definition 1 (Relations). *A* RuleML relation *is a named predicate or a named builtin predicate:*

$\langle \mathit{Rel} \rangle$ ::= **Rel(**$\langle \mathit{relID} \rangle$**).**

$\langle \mathit{relID} \rangle$ ::= **URIreference.**

In this paper we follow the Datalog (constructor-function-free) sublanguage, the foundation for the kernel of RuleML, i.e. we don't not consider functions in atom construction (Datalog model).

Definition 2 (Terms). *A RuleML* term *is a* variables *or an* individual constant*:*

$\langle \mathit{Term} \rangle$ ::= $\langle \mathit{Variable} \rangle$ | $\langle \mathit{Individual} \rangle$**.**

$\langle \mathit{Variable} \rangle$ ::= **Var(**$\langle \mathit{varName} \rangle$**).**

$\langle \mathit{Individual} \rangle$ ::= **Ind(**$\langle \mathit{individualID} \rangle$**).**

$\langle \mathit{individualID} \rangle$ ::= **URIreference.**

$\langle \mathit{varName} \rangle$ ::= **UnicodeString.**

Slots are basic RuleML constructs which are used in slotted atoms (see below) for capturing object descriptions.

Definition 3 (Slots). *The abstract syntax of* RuleML slots *is:*

$\langle \mathit{Slot} \rangle$ ::= **Slot(Property(**$\langle \mathit{slID} \rangle$**),** $\langle \mathit{slotValue} \rangle$**).**

$\langle \mathit{slotValue} \rangle$ ::= $\langle \mathit{Individual} \rangle$ | $\langle \mathit{Data} \rangle$

$\langle \mathit{Data} \rangle$ ::= **Data(**$\langle \mathit{dataValue} \rangle$**).**

$\langle \mathit{dataValue} \rangle$ ::= **UnicodeString.**

$\langle \mathit{slID} \rangle$ ::= **URIreference.**

RuleML allow two kinds of atoms: *positional atoms* and *slotted atoms.* Positional atoms are used for representing Prolog like atoms while the slotted atoms capture object descriptions.

Definition 4 (Atoms). *The abstract syntax of* RuleML atoms *is:*

$\langle \mathit{Atom} \rangle$ ::= $\langle \mathit{PositionalAtom} \rangle$ | $\langle \mathit{SlottedAtom} \rangle$**.**

$\langle \mathit{PositionalAtom} \rangle$ ::= **PosAtom(**$\langle \mathit{Rel} \rangle$**, {**$\langle \mathit{Term} \rangle$**}).**

$\langle \mathit{SlotAtom} \rangle$ ::= **SlotAtom(**$\langle \mathit{Term} \rangle$**, {**$\langle \mathit{Slot} \rangle$**}).**

Below we provide the abstract syntax of literals (negated atoms) in RuleML. Intuitively speaking, *weak negation* captures the absence of positive information, while *strong negation* captures the presence of explicit negative information (in the sense of Kleene's 3-valued logic).

Definition 5 (Negations). *The abstract syntax of weak negation and strong negations is:*

$\langle \textit{WeakNegation} \rangle$ ::= **Naf(**$\langle \textit{Atom} \rangle$**).**

$\langle \textit{StrongNegation} \rangle$ ::= **Neg(**$\langle \textit{Atom} \rangle$**).**

Definition 6 (AndOrNafNegFormula). *The abstract syntax of* RuleML AndOrNafNegFormula *is:*

$\langle \textit{AndOrNafNegFormula} \rangle$ ::= $\langle \textit{Atom} \rangle$ | $\langle \textit{WeakNegation} \rangle$ | $\langle \textit{StrongNegation} \rangle$
| $\langle \textit{Conjunction} \rangle$ | $\langle \textit{Disjunction} \rangle$

$\langle \textit{Conjunction} \rangle$ ::= **And({**$\langle \textit{AndOrNafNegFormula} \rangle$**}).**

$\langle \textit{Disjunction} \rangle$ ::= **Or({**$\langle \textit{AndOrNafNegFormula} \rangle$**}).**

A RuleML *rule* has a *body* which is represented by an and-or-naf-neg-formula and a *head* which is an atom or a strong negated atom. Rules with an empty body are *facts* and rules with an empty head are *queries.*

Definition 7 (Rule). *The abstract syntax of the rule in RuleML is:*

$\langle \textit{Rule} \rangle$ ::= **Imp(**[$\langle \textit{ruleID} \rangle$,][$\langle \textit{Body} \rangle$,][$\langle \textit{Head} \rangle$]**).**

$\langle \textit{Body} \rangle$ ::= **Body(**[$\langle \textit{AndOrNafNegFormula} \rangle$]**).**

$\langle \textit{Head} \rangle$ ::= **Head(**$\langle \textit{Atom} \rangle$**)** | **Head(**$\langle \textit{StrongNegation} \rangle$**).**

$\langle \textit{ruleID} \rangle$ ::= **URIreference.**

In the following example presents the abstract syntax form of a rule from the RDF/RuleML interoperability position paper [3].

Example 1 (A rule). The following is an example of a rule using both kinds of negation. Here, the relation requiresService does not allow Closed-World inference, while the relation isAssignedToRentalContract does allow it, and, hence, the former is negated with Neg and the latter with Naf.

```
Imp("R1",
 Body(
  And(
   PosAtom(Rel(ex:RentalCar),Var(Car))
   Neg(
    PosAtom(Rel(ex:requiresService),Var(Car))
   )
   Naf(
    PosAtom(Rel(isAssignedToRentalContract),Var(Car))
   )
  )
 )
 Head(
  PosAtom(Rel(ex:isAvailable),Var(Car))
 )
)
```

Definition 8 (Knowledge Base). *Any* RuleML **Knowledge Base** *is a set of rules.*

$\langle KB \rangle ::= \{\langle Rule \rangle\}$.

3 RuleML Semantics in Partial Logic

In business and administrative domains, most of the information to be processed is assumed to be complete (this tacit assumption is called Closed-World Assumption, [8] in AI). But in other domains, such as in medicine and criminology, most information is incomplete. RuleML is a rule language for applications so it should allow expressing rules for both types of information.

3.1 Why Partial Logic?

Unlike negation in classical logic, real-world negation is not a simple two-valued truth function. The simplest generalization of classical logic that is able to account for two kinds of negation is partial logic giving up the classical bivalence principle and subsuming a number of 3-valued and 4-valued logics (see [7]).

Because the web does not operate under de CWA hypothesis (see [1]), RuleML should distinguish between relations (or slots) that are totaly represented (a total assumption corresponds to a predicate-specific Closed-World Assumption) or are partial represented.

In the case of a completely represented predicate, negation-as-failure reflects falsity, and negation-as-failure and strong negation collapse into classical negation. In the case of a partial represented relation or slot, negation-as-failure only reflects non-provability, but does not allow to infer the classical negation (for details see [11]).

In standard logics, there is a close relationship between a derivation rule and the corresponding implicational formula: they have the same models. For nonmonotonic rules (with negation-as-failure, [4]) this is no longer the case: the intended models of such a rule are in general not the same as the intended models of the corresponding implication (see [6] and [5]).

These are the main reasons for that we consider partial logic to be the most appropriate logic for interpreting RuleML rules.

3.2 Direct-Model Theoretic Semantics

Definition 9 (Vocabulary). *The* RuleML vocabulary *is defined by the following tuple*

$$\mathcal{V}oc = (\mathrm{Rel}, \mathrm{TRel}, \mathrm{Pr}, \mathrm{DLit}, \mathrm{Var}, \mathrm{Ind})$$

where Rel *is the set of relation names,* TRel *is the set of total relations names,* Pr *is the set of property names,* DLit *is the set of all data literals names,* Var *is the set of variables names and* Ind *is the set of individuals names.*

Note that Pr *is in fact the set of al URI references to binary relations, i.e.* $\mathrm{Pr} \subseteq \mathrm{Rel}$. *We suppose that all these sets are nonempty, pairwise disjoint and* $\mathrm{TRel} \subseteq \mathrm{Rel}$.

Let $\mathcal{O}$ be the set of all objects and $\mathcal{V}(DLit)$ the set of all data literal values. In this paper each relation or property is a pair $r = \langle r^t, r^f \rangle$ such that $r^t, r^f \subseteq \mathcal{O}^2$. We allow only coherent relations and properties (see [7] for details), i.e, if r is a relation, then $r^t \cap r^f = \emptyset$.

Definition 10 (Interpretation). *An abstract interpretation is a tuple of functions*

$$\mathcal{I} = (\mathcal{I}_{DLit}, \mathcal{I}_{Ind}, \mathcal{I}_{Pr}, \mathcal{I}_{Rel})$$

such that:

1. $\mathcal{I}_{DLit} : \mathrm{DLit} \longrightarrow \mathcal{V}(DLit)$*, maps each data literal into a value i.e.*

$$\mathcal{I}(Data(d)) = \mathcal{I}_{DLit}(d) \in \mathcal{V}(DLit)$$

   where $\mathcal{V}(DLit)$ *is the value space*[3].
2. $\mathcal{I}_{Ind} : \mathrm{Ind} \longrightarrow \mathcal{O}$*, maps individuals names to objects,*

$$\mathcal{I}(Ind(id)) = \mathcal{I}_{Ind}(id) \in \mathcal{O}$$

3. $\mathcal{I}_{Pr} : \mathrm{Pr} \longrightarrow \mathcal{O}^2 \times \mathcal{O}^2$*, maps each property name into a pair of binary relations,*

$$\mathcal{I}(Pr(p)) = \mathcal{I}_{Pr}(p) = \langle p^t, p^f \rangle, \text{ with } p^t, p^f \subseteq \mathcal{O}^2$$

   If p *is a total property name, then* $p^t \cup p^f = \mathcal{O}^2$*. Note that in this paper we allow only coherent properties (see [7] for details) i.e.* $p^t \cap p^f = \emptyset$.
4. $\mathcal{I}_{Rel} : \mathrm{Rel} \longrightarrow \mathcal{O}^n \times \mathcal{O}^n$*, maps each n-ary relation name into a pair of n-ary relations*

$$\mathcal{I}(Rel(r)) = \mathcal{I}_{Rel}(r) = \langle r^t, r^f \rangle, \text{ with } r^t, r^f \subseteq \mathcal{O}^n$$

   If r *is a total n-ary relation name, then* $r^t \cup r^f = \mathcal{O}^n$*. Again we allow only coherent relations ([7]) i.e.* $r^t \cap r^f = \emptyset$.

[3] As in RDF, a datatype is characterized by a lexical space, DLit which is a set of Unicode strings; a value space, $\mathcal{V}(Dlit)$; and a total mapping $\mathcal{I}_{DLit}$ from the lexical space to the value space.

Definition 11 (Valuation). *A valuation over an interpretation $\mathcal{I}$ is a function $\mathcal{V} : \mathrm{Var} \longrightarrow \mathcal{O}$, which associate each variable name with an object, i.e.*

$$\mathcal{I}(Var(v)) = \mathcal{V}(v) \in \mathcal{O}$$

Definition 12 (Satisfaction Relation). *Let $\mathcal{V}$ be a valuation. The* satisfaction relation $\models$ *is a pair $\langle \models^t, \models^f \rangle$ such that,*

1. *Let r be a relation name and $\mathcal{I}_{Rel}(r) = (r^t, r^f)$.*
   (a) If r is partial, i.e. $r \in \mathrm{Rel} - \mathrm{TRel}$, then

$$\begin{array}{l} \mathcal{I},_{\mathcal{V}} \models^t PosAtom(r, t_1, \ldots, t_n) \ iff \ \langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \in r^t \\ \mathcal{I},_{\mathcal{V}} \models^f PosAtom(r, t_1, \ldots, t_n) \ iff \ \langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \in r^f \end{array}$$

   (b) If r is total, i.e. $r \in \mathrm{TRel}$, then

$$\begin{array}{l} \mathcal{I},_{\mathcal{V}} \models^t PosAtom(r, t_1, \ldots, t_n) \ iff \ \langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \in r^t \\ \mathcal{I},_{\mathcal{V}} \models^f PosAtom(r, t_1, \ldots, t_n) \ iff \ \langle \mathcal{I}(t_1), \ldots, \mathcal{I}(t_n) \rangle \notin r^t \end{array}$$

2. *Let $(p_i)_{i=1,n}$ be property names, i.e. $\mathcal{I}_{Pr}(p_i) = (p_i^t, p_i^f)_{i=1,n}$, t be a term and $S = \{Slot(Property(p_i), v_i)\}_{i=1,n}$ a slot. Then,*
   (a)

$$\mathcal{I},_{\mathcal{V}} \models^t SlotAtom(t, S) \ iff \bigwedge_{i=1,n} \{\langle \mathcal{I}(t), \mathcal{I}(v_i) \rangle \in p_i^t\}$$

   (b)

$$\mathcal{I},_{\mathcal{V}} \models^f SlotAtom(t, S)$$

$$iff$$

$$(\exists i,\ p_i\ total,\ \langle \mathcal{I}(t), \mathcal{I}(v_i) \rangle \notin p_i^t) \vee (\exists j,\ p_j\ partial,\ \langle \mathcal{I}(t), \mathcal{I}(v_j) \rangle \in p_j^f)$$

3. *If $R = Imp(Body(P_1, \ldots, P_n)), Head(C))$ is a rule, then*

$$\mathcal{I},_{\mathcal{V}} \models^t Head(C) \ if \ \mathcal{I} \models^t (P_1 \wedge \ldots \wedge P_n)$$

Definition 13 (Model and Satisfaction Set). *Let R be a rule in RuleML. We say that an interpretation $\mathcal{I} = (\mathcal{I}_{DLit}, \mathcal{I}_{Ind}, \mathcal{I}_{Pr}, \mathcal{I}_{Rel})$ is a* model for *R and we denote $\mathcal{I} \models R$ if and only if*

$$\mathcal{I},_{\mathcal{V}} \models^t \ R$$

for all variable valuations $\mathcal{V}$.

Let $KB = \{R_1, \ldots, R_n\}$ be a RuleML knowledge base. A satisfaction set *for KB is the set:*

$$Sat_{\mathcal{I}}(KB) = \{\mathcal{V} \mid \mathcal{I},_{\mathcal{V}} \models^t (R_1 \wedge \ldots \wedge R_n)\}$$

Definition 14 (Rule Model). *Let $\mathcal{I}$ be an interpretation and $R = Imp(Body(P_1, \ldots, P_n)), Head(C))$ a rule.*

$$\mathcal{I} \models R \ iff \bigcap_{i \leq n} Sat_{\mathcal{I}}(P_i) \subseteq Sat_{\mathcal{I}}(C)$$

Example 2. Let $\mathcal{V}oc$ = (Rel, TRel, Pr, DLit, Var, Ind) be the following vocabulary:

$$\begin{aligned}&\text{Rel} = \{\text{"}ex : RentalCar\text{"}, \text{"}ex : requiresService\text{"},\\ &\text{"}ex : isAssignedToRentalContract\text{"}, \text{"}ex : isAvailable\text{"}\}\\ &\text{TRel} = \{\text{"}ex : RentalCar\text{"}, \text{"}ex : isAssignedToRentalContract\text{"}\}\\ &\text{Pr} = \emptyset\\ &\text{DLit} = \emptyset\\ &\text{Var} = \{Car\}\\ &\text{Ind} = \{\text{"}ex : CT2MDF\text{"}, \text{"}ex : DJ02GCP\text{"}\}\end{aligned}$$

Let $KB = \{R1, R2, R3, R4\}$ be a RuleML knowledge base which consist of the rule $R1$ as in Example 1 and the following facts (rules with empty body):

1. Rule $R2$ "ex:CT20MDF is a RentalCar".
2. Rule $R3$ "ex:DJ02GCP is a RentalCar".
3. Rule $R4$ "ex:DJ02GCP does not require service" ("requiredService is a partial relation").

Below is the abstract semantics of the new rules:

```
Imp("R2",
 Head(PosAtom(Rel("ex:RentalCar"),Ind("ex:CT20MDF")))
)
Imp("R3",
 Head(PosAtom(Rel("ex:RentalCar"),Ind("ex:DJ02GCP")))
)
Imp("R4",
 Head(
  Neg(
   PosAtom(Rel("ex:requireService"),Ind("ex:DJ02GCP"))
  )
 )
)
```

In the following we define an interpretation $\mathcal{I} = (\mathcal{I}_{DLit}, \mathcal{I}_{Ind}, \mathcal{I}_{Sl}, \mathcal{I}_{Rel})$ and the we illustrate some satisfaction results in KB.

1. $\mathcal{I}_{DLit}$ can be any arbitray function (we don't use this function because don't have data literals in our knowledge base).
2. $\mathcal{I}_{Ind}$ is defined below (for simplicity we keep the name of the individual removing the namespace):

$$\begin{aligned}&\mathcal{I}_{Ind}(\text{"}ex : CT20MDF\text{"}) = \text{"}Opel - Corsa\text{"}\\ &\mathcal{I}_{Ind}(\text{"}ex : DJ02GCP\text{"}) = \text{"}BMW6\text{"}\end{aligned}$$

3. $\mathcal{I}_{Sl}$ can be any arbitray function (again, we don't use this function because don't have slot relations in our knowledge base).

4. $\mathcal{I}_{Rel}$ is defined below:

$$\mathcal{I}_{Rel}(ex : RentalCar) = (rc^t, rc^f)$$
$$\mathcal{I}_{Rel}(ex : requireService) = (rs^t, rs^f)$$
$$\mathcal{I}_{Rel}(ex : isAssignedToRentalContract) = (irc^t, irc^f)$$
$$\mathcal{I}_{Rel}(ex : isAvailable) = (ia^t, ia^f)$$

where

$rc^t = \{Ind(''ex : CT20MDF''), Ind(''ex : DJ02GCP'')\}$, $rc^f = \emptyset$,
$rs^t = \emptyset$, $rs^f = \{Ind(''ex : DJ02GCP'')\}$,
$irc^t = \{Ind(''ex : CT20MDF'')\}$, $irc^f = \emptyset$,
$ia^t = \emptyset$, $ia^f = \emptyset$.

Now, let $\mathcal{V}_1 : \text{Var} \longrightarrow \mathcal{O}$, $\mathcal{V}_1(Car) = ''DJ02GCP''$. Then,

$$\mathcal{I}, _{\mathcal{V}_1} \models^t PosAtom(Rel(ex : isAvailable), Ind(''ex : DJ02GCP''))$$

that means "the rental car BMW6 is available for renting".

$$\mathcal{I}, _{\mathcal{V}_1} \models^f PosAtom(Rel(ex : isAvailable), Ind(''ex : CT20MDF''))$$

that means "the rental car Opel Corsa is not available for renting".

4 Minimal Reasoning

Let $\mathcal{V}oc = (\text{Rel}, \text{TRel}, \text{Pr}, \text{DLit}, \text{Var}, \text{Ind})$ be a vocabulary and KB a RuleML knowledge base based on $\mathcal{V}oc$.

We denote by $Lit_{KB}(\text{Ind})$ the set of all atoms or strong negation of atoms without variables that are constructed using atoms from KB and indivivual constants from Ind i.e the Herbrand base of KB. Each $\mathcal{I}^H \subseteq Lit_{KB}(\text{Ind})$ is a Herbrand interpretation for KB.

Definition 15. *If $\mathcal{I} \in Lit_{KB}(\text{Ind})$ such that $\mathcal{I} \models KB$ we say $\mathcal{I}$ is a* Herbrand model of KB. *Usually we denote a Herbrand interpretation by $\mathcal{I}^H$, a Herbrand model by $\mathcal{M}^H$ and the set of all Herbrand models by $\mathcal{M}od^H(KB)$.*

Definition 16. *Let $\mathcal{I}_1^H, \mathcal{I}_2^H \subseteq Lit_{KB}(\text{Ind})$ be two Herbrand interpretations. We say that $\mathcal{I}_2^H$* extends *$\mathcal{I}_1^H$ and denote $\mathcal{I}_1^H \preceq \mathcal{I}_2^H$, if*

$$\{L \in Lit_{KB}(\text{Ind}) \mid \mathcal{I}_1^H \models L\} \subseteq \{L \in Lit_{KB}(\text{Ind}) \mid \mathcal{I}_2^H \models L\}$$

The informal meaning of this order is: $\mathcal{I}_1^H$ is "less informative than" $\mathcal{I}_2^H$.

Definition 17 (Interpretation Intervals). *The model* interval *is defined as:*

$$[\mathcal{I}_1^H, \mathcal{I}_2^H] = \{\mathcal{I}^H \in Lit_{KB}(\text{Ind}) \mid \mathcal{I}_1^H \preceq \mathcal{I}^H \preceq \mathcal{I}_2^H\}$$

Definition 18 (Stable Model ([7], [13])). *Let $\mathcal{M}^H \in \mathcal{M}od^H(KB)$ be a Herbrand model. $\mathcal{M}^H$ is called* minimally stable model of *KB if there is a chain of Herbrand interpretations $\mathcal{M}_0^H \preceq \ldots \preceq \mathcal{M}_k^H$ such that $\mathcal{M}^H = \mathcal{M}_k^H$ and:*

1. $\mathcal{M}_0^H = \emptyset$.
2. For all $i \in \{1, \ldots, k\}$, $\mathcal{M}_i^H$ is a minimal extension of $\mathcal{M}_{i-1}^H$ satisfying the heads of all rules whose bodies hold in $\left[\mathcal{M}_{i-1}^H, \mathcal{M}^H\right]$, i.e.

$$\mathcal{M}_i^H \in \min\left\{\mathcal{I}^H \mid \mathcal{M}_{i-1}^H \preceq \mathcal{I}^H \text{ and } \mathcal{I}^H \models H(R), \text{ for all } R \in \mathcal{R}_{\left[\mathcal{M}_{i-1}^H, \mathcal{M}^H\right]}\right\}$$

where R is a rule, $H(R)$ is the head of rule R, $B(R)$ is the body of R and

$$\mathcal{R}_{\left[\mathcal{M}_{i-1}^H, \mathcal{M}^H\right]} = \left\{R \in KB \mid \mathcal{M}^H \models B(R) \text{ for all } \mathcal{M}^H \in \left[\mathcal{M}_{i-1}^H, \mathcal{M}^H\right]\right\}$$

Example 3. (continued) Let $KB = \{R1, R2, R3, R4\}$ be a RuleML knowledge base as in Example 2.

Let's test if the following model $\mathcal{M} = \{L_1, L_2, L_3, L_4\}$ where

$L_1 = PosAtom(Rel(ex : isAvailable), Ind("ex : CT20MDF"))$
$L_2 = PositionalAtom(Rel("ex : RentalCar"), Ind("ex : CT20MDF"))$
$L_3 = PositionalAtom(Rel("ex : RentalCar"), Ind("ex : DJ02GCP"))$
$L_4 = Neg(PositionalAtom(Rel("ex : requireService"), Ind("ex : DJ02GCP")))$

is a stable model of KB.
Let $\mathcal{M}_0^H = \emptyset$. Then $\mathcal{R}_{\left[\mathcal{M}_0^H\right]} = \{R_2, R_3, R_4\}$ and, consequently,

$$\mathcal{M}_1^H = \{L_2, L_3, L_4\}$$

Now, $\mathcal{R}_{\left[\mathcal{M}_0^H, \mathcal{M}_1^H\right]} = \{R_1, R_2, R_3, R_4\}$ and

$$\mathcal{M}_2^H = \{L_1, L_2, L_3, L_4\}$$

Finally, $\mathcal{R}_{\left[\mathcal{M}_1^H, \mathcal{M}_2^H\right]} = \{R_1, R_2, R_3, R_4\}$ and

$$\mathcal{M}_3^H = \mathcal{M}_2^H$$

5 Conclusions and Future Work

The paper provide an abstract syntax for the core RuleML sublanguage NafNeg-Datalog and a model theoretic semantic. The semantics is done using partial model theory, as a natural generalization of classical model theory. This semantics is able to capture many important distinctions arising in web rules reasoning, such as explicit falsity vs. non-truth, or total vs. partial predicates. At the object level, these distinctions can be expressed by means of the two negations of partial logic.

One possible quick extension of this work is to generalize the strong negation formulas that is to allow strong negations on and-or-naf-neg-formulas not only to atoms like below:

Example 4 (Allowing strong negation on and-or-naf-neg-formulas).

⟨*StrongNegation*⟩ ::= **Neg(**⟨*AndOrNafNegFormula*⟩**).**

All cases of a such formula composition are treated by the following DeMorgan-style rewrite rules expressing the falsification of compound formulas:

$$\begin{array}{l} Neg(F \wedge G) \rightarrow Neg(F) \vee Neg(G) \\ Neg(F \vee G) \rightarrow Neg(F) \wedge Neg(G) \\ Neg(Neg(F)) \rightarrow F \\ Neg(Naf(A)) \rightarrow A \end{array}$$

where F,G are and-or-naf-neg-formulas and A is an atom.

Another extension concern the improvement of the abstract syntax to allow the declaration of a total or partial relation/slot like:

Example 5 (Declare total or partial relations/slots).

⟨*Rel*⟩ ::= **Rel(**⟨*relID*⟩, ⟨*type*⟩**).**

⟨*SlotAtom*⟩ ::= **SlotAtom(**⟨*Term*⟩, ⟨*type*⟩, **{**⟨*Slot*⟩**}).**

⟨*type*⟩ ::= `total` | `partial`.

This improvement will be used by RuleML reasoning engines.

Using partial logic we solve the knowledge representation which need incomplete (partial) predicates. However, it is possible to extend RuleML with capabilities for representing uncertain information in the form of fuzzy sets, fuzzy numbers, possibility and necessity measures, discrete plausibility measures or other quantitative measures of uncertainty.

Also, we don't cover in this paper the important problems of non-coherent relations or non-coherent slots that must be also declared by the rule designer. Allowing non coherent relations and non-coherent slots means to accommodate an inconsistency tolerant reasoning in RuleML.

Finally, seems to be possible to extend RuleML to accommodate the distinction between OWL individuals, instead of using the traditional logic concept individual constants.

References

1. Berners Lee, T., Hendler J., Lassila, O., The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American May 2001.
2. Boley, H., Tabet, S., Wagner, G., Design Rationale of RuleML:A Markup Language for Semantic Web Rules, in Proc. of Int. Semantic Web Working Symposium (SWWS), July 30 - August 1, 2001, Stanford University, California, USA.
3. Boley, H., Mei, J., Sintek, M., Gerd Wagner, G., RDF/RuleML Interoperability, W3C Workshop on Rule Languages for Interoperability Position Paper: 27-28 April 2005, available at http://www.w3.org/2004/12/rules-ws/paper/93/

4. Clark, K.L., Negation as Failure, In H. Gallaire and J. Minker Editors, Logic and Databases, pp.293-322, Plenum Press, 1978.
5. van Gelder, A., Kenneth A. Ross A., K., Schlipf S., J., The Well-Founded Semantics for General Logic Programs, Journal of ACM, Vol.38, No. 3 July 1991, pp.620-650.
6. Gelfond, M., Lifschitz, V., The Stable Model Semantics For Logic Programming, In Proceedings of the 5th International Conference on Logic Programming, 1070–1080. Seattle, USA, August 1988. The MIT Press.
7. Herre, H., Jaspars, J., Wagner G., Partial Logics with Two Kinds of Negation as a Foundation for Knowledge-Based Reasoning, in D.M. Gabbay and H. Wansing (Eds.), What is Negation ?, Kluwer Academic Publishers, 1999.
8. Reiter, R., On Closed World Data Bases, In H. Gallaire and J. Minker Editors, Logic and Databases, pp. 55-76, Plenum Press, 1978.
9. Wagner, G., Antoniou, G., Tabet, S., and Boley, H.,The Abstract Syntax of RuleML Towards a General Web Rule Language Framework, Rule Markup Initiative (RuleML), http://www.RuleML.org
10. Wagner, G., Web Rules Need Two Kind of Negations, in Proc. of Principles and Practice of Semantic Web Reasoning, PPSWR 2003, pp.33-50.
11. Wagner, G., Foundations of Knowledge Systems with Applications to Databases and Agents, Kluwer Academic Publishers, 1998.
12. Wagner, G., Seven Golden Rules for a Web Rule Language, invited contribution to the Trends & Controversies section of IEEE Intelligent Systems 18:5, Sept/Oct 2003.
13. Wagner, G., Herre, H., Stable Models are Generated by a Stable Chain. Journal of Logic Programming 30 (1997) 2, 165 - 177.