

# Semantic Web

## Rule Systems

F. Abel, N. Henze, D. Krause

IVS Semantic Web Group

8.1.2009

## 1 Rule Systems

- Motivation
- Rules
- Facts and Logic Programs
- Goals

## 2 Monotonic Rule Systems

- Definition
- Example

## 3 Non-monotonic Rule Systems

- Motivation
- Defeasible Rules
- Example

## 4 Rule Markup

- Rule Markup in XML: Monotonic Rules
- Rule Markup in XML: Non-monotonic Rules
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML

## 1 Rule Systems

- Motivation
- Rules
- Facts and Logic Programs
- Goals

## 2 Monotonic Rule Systems

- Definition
- Example

## 3 Non-monotonic Rule Systems

- Motivation
- Defeasible Rules
- Example

## 4 Rule Markup

- Rule Markup in XML: Monotonic Rules
- Rule Markup in XML: Non-monotonic Rules
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML

- Rules for modeling applications' processes
- Rules for modeling applications' constraints
- Examples for rules:
  - Example of Lucy and Pete
  - Business Rules

Slides are based on: G. Antoniou and F. van Harmelen: A Semantic Web Primer.

## Definition (Rules)

A rule has the form

$B_1, \dots, B_n, \longrightarrow A,$  where  $A, B_i$  are atomic formulas.

$A$  is the **head** — **consequent** of the rule

$B_1, \dots, B_n$  are the **body** — **premises** — **antecedent** (Prämissen, Voraussetzung).

The commas in the premise denote conjunction.

## Example (Syntax of Rules)

$loyalCustomer(X), age(X) > 60 \longrightarrow discount(X)$

**variables** placeholders for values, e.g.  $X$

**constants** denote fixed value, e.g. 60

**predicates** relate objects: e.g.  $loyalCustomer, >$

**function symbols** return a value for certain arguments, e.g.  $age$

## Definition (Rules (Continued))

All variables occurring in a rule are universally quantified. This means: A rule can be interpreted as formula::

$$\forall X_1 \dots \forall X_k ((B_1 \wedge \dots \wedge B_n) \longrightarrow A)$$

or equivalent

$$\forall X_1 \dots \forall X_k (A \vee \neg B_1 \vee \dots \vee \neg B_n)$$

where  $X_1 \dots X_k$  are all variables occurring in  $A, B_1, \dots, B_n$ .

*Remark: A **Horn clause** is a disjunction of literals of which at most one is positive.*

## Definition (Fact)

A fact is an atomic formula.

## Example (Fact)

*loyalCustomer(a3456789)* is an atomic formula; it says that the customer with ID a3456789 is loyal.

## Definition (Logic Program)

A logic program is a finite set of facts and rules.

## Definition (Goal)

A goal denotes a query  $Q$  asked to a logic program. It has the form

$$B_1, \dots, B_n \longrightarrow$$

For  $n = 0$  we have the empty goal.

- equivalent formulation of a goal:

$$\forall X_1, \dots, X_k (\neg B_1 \vee \dots \vee \neg B_n)$$

where  $X_1 \dots X_k$  are all variables occurring in  $B_1, \dots, B_n$ .

- again equivalent:

$$\neg \exists X_1, \dots, X_k (B_1 \wedge \dots \wedge B_n)$$

- to proof a goal: **proof by contradiction**: We proof that a goal can be answered positively by negating the goal and proving that we get a contradiction using the logic program

- 1 Rule Systems
  - Motivation
  - Rules
  - Facts and Logic Programs
  - Goals
- 2 Monotonic Rule Systems
  - Definition
  - Example
- 3 Non-monotonic Rule Systems
  - Motivation
  - Defeasible Rules
  - Example
- 4 Rule Markup
  - Rule Markup in XML: Monotonic Rules
  - Rule Markup in XML: Non-monotonic Rules
  - SWRL: A Semantic Web Rule Language Combining OWL and RuleML

## Definition (Monotonic Rule System)

If a conclusion can be drawn, it remains valid even if new knowledge becomes available.

## Example (Monotonic Rules: Family Relationships)

Facts:

*mother*( $X, Y$ )       $X$  is mother of  $Y$

*father*( $X, Y$ )       $X$  is father of  $Y$

*male*( $X$ )           $X$  is male

*female*( $X$ )         $X$  is female

Rules for deriving further relationships:

*mother*( $X, Y$ )  $\longrightarrow$  *parent*( $X, Y$ )

*father*( $X, Y$ )  $\longrightarrow$  *parent*( $X, Y$ )

## Example (Monotonic Rules: Family Relationships (Ctd.))

A brother is a male person sharing a parent:

$$\text{male}(X), \text{parent}(P, X), \text{parent}(P, Y), \text{notSame}(X, Y) \longrightarrow \text{brother}(X, Y)$$

An uncle is a brother of a parent

$$\text{brother}(X, P), \text{parent}(P, Y) \longrightarrow \text{uncle}(X, Y)$$

A grandmother is the mother of a parent:

$$\text{mother}(X, P), \text{parent}(P, Y) \longrightarrow \text{grandmother}(X, Y)$$

An ancestor is either a parent or an ancestor of a parent

$$\text{parent}(X, Y) \longrightarrow \text{ancestor}(X, Y)$$

$$\text{ancestor}(X, P), \text{parent}(P, Y) \longrightarrow \text{ancestor}(X, Y)$$

## 1 Rule Systems

- Motivation
- Rules
- Facts and Logic Programs
- Goals

## 2 Monotonic Rule Systems

- Definition
- Example

## 3 Non-monotonic Rule Systems

- Motivation
- Defeasible Rules
- Example

## 4 Rule Markup

- Rule Markup in XML: Monotonic Rules
- Rule Markup in XML: Non-monotonic Rules
- SWRL: A Semantic Web Rule Language Combining OWL and RuleML

## Example (Why do we need non-monotonic rule systems?)

$R1$  : If birthday, then special discount

$R2$  : If not birthday, then no special discount

- works well if birthday is known, this means where we have complete information (its known whether it is birthday or not)
- otherwise we might think of something like

$R1$  : If birthday, then special discount

$R2'$  : If birthday is not known, then no special discount

- if we get new information on the birthday, we would like to eventually offer a discount!

- **Defeasible Rules / Anfechtbare Regeln:**

- A rule may be defeated by others
- To allow conflict between rules, negated atomic formulas may occur in the head and the body of rules, e.g.

$$\begin{aligned} p(X) &\implies q(X) \\ r(X) &\implies \neg q(X) \end{aligned}$$

- to distinguish the non-monotonic rules from the monotonic rules, we use  $\implies$  instead of  $\longrightarrow$ .
- **Priorities to resolve conflicts**, e.g.

- external priority relations

$$\begin{aligned} r_1 : p(X) &\implies q(X) \\ r_2 : r(X) &\implies \neg q(X) \\ \text{and } r_1 &> r_2 \end{aligned}$$

- condition on an external priority relation: acyclic.

## Definition (Defeasible Rule)

A defeasible rule has the form

$$r : L_1, \dots, L_n \Longrightarrow L$$

where  $r$  is the **label**,  $L_1, \dots, L_n$  is the **body** / **premises**, and  $L$  is the **head** of the rule.

$L, L_1, \dots, L_n$  are positive or negative literals ( a literal is an atomic formula  $P(t_1, \dots, t_n)$  or its negation).

## Definition (Defeasible Logic Program)

A defeasible logic program is a triple

$$(F, R, >)$$

consisting of a set  $F$  of facts, a finite set  $R$  of defeasible rules, and an acyclic binary relation  $>$  on  $R$ .

## Example (e-Commerce: Brokered Trade)

- broker: independent, third party which proposes transactions
- broker matches the buyer's requirements and the sellers' capabilities

Buyer's requirements:

*Carlos is looking for an apartment of at least 45 m<sup>2</sup> with at least two bedrooms. If it is on the third floor or higher, the house must have an elevator. Also, pets must be allowed.*

*Carlos is willing to pay \$300 for a centrally located 45 m<sup>2</sup> apartment, and \$250 for a similar flat in the suburbs. In addition, he is willing to pay an extra \$5 per square meter for a larger apartment, and \$2 per square meter for a garden.*

*He is unable to pay more than \$400 in total. If given the choice, he would go for the cheapest option. His second priority is the presence of a garden; his lowest priority is additional space.*

# Non-monotonic Rule Systems - Example II

Formalization:

<i>size</i> (X, Y)	Y is the size of apartment X (in sq m)
<i>bedrooms</i> (X, Y)	X has Y bedrooms
<i>price</i> (X, Y)	X has the price Y
<i>floor</i> (X, Y)	X is on the Yth floor
<i>gardenSize</i> (X, Y)	X has a garden of size Y
<i>lift</i> (X)	X has a lift
<i>pets</i> (X)	pets are allowed in X
<i>central</i> (X)	X is centrally located

We use the following predicates:

<i>acceptable</i> (X)	flat X satisfies Carlos' requirements
<i>offer</i> (X, Y)	for X, Carlos is willing to pay Y \$

# Non-monotonic Rule Systems - Example III

Firm requirements: Any requirement is a priori acceptable

$$r_1 : \quad \Longrightarrow \text{ acceptable}(X)$$

It is unacceptable if one of Carlos' requirements is not met:

$$r_2 : \quad \text{bedrooms}(X, Y), Y < 2 \quad \Longrightarrow \quad \neg \text{acceptable}(X)$$

$$r_3 : \quad \text{size}(X, Y), Y < 45 \quad \Longrightarrow \quad \neg \text{acceptable}(X)$$

$$r_4 : \quad \neg \text{pets}(X) \quad \Longrightarrow \quad \neg \text{acceptable}(X)$$

$$r_5 : \quad \text{floor}(X, Y), Y > 2, \neg \text{lift}(X) \quad \Longrightarrow \quad \neg \text{acceptable}(X)$$

$$r_6 : \quad \text{price}(X, Y), Y > 400 \quad \Longrightarrow \quad \neg \text{acceptable}(X)$$

Rules  $r_2, \dots, r_6$  are exceptions to rule  $r_1$ :

$$r_2 > r_1, r_3 > r_1, r_4 > r_1, r_5 > r_1, r_6 > r_1.$$

# Non-monotonic Rule Systems - Example III

Next we calculate the price Carlos is willing to pay:

$$r_7 : \quad \text{size}(X, Y), Y \geq 45, \text{garden}(X, z), \text{central}(X) \\ \implies \text{offer}(X, 300 + 2z + 5(Y - 45))$$

$$r_8 : \quad \text{size}(X, Y), Y \geq 45, \text{garden}(X, z), \neg \text{central}(X) \\ \implies \text{offer}(X, 250 + 2z + 5(Y - 45))$$

And

$$r_7 > r_1, r_8 > r_1.$$

An apartment is only acceptable if the amount Carlos is willing to pay is not less than the price specified by the landlord

$$r_9 : \quad \text{offer}(X, Y), \text{price}(X, z), Y < z \implies \neg \text{accetable}(X) \\ \text{and } r_9 > r_1.$$

# Non-monotonic Rule Systems - Example IV

Flat	Bedrooms	size	Central	Floor	Lift	Pets	Garden	Price
$a_1$	1	50	yes	1	no	yes	0	300
$a_2$	2	45	yes	0	no	yes	0	335
$a_3$	2	65	no	2	no	yes	0	350
$a_4$	2	55	no	1	yes	no	15	330
$a_5$	3	55	yes	0	no	yes	15	350
$a_6$	2	60	yes	3	no	no	0	370
$a_7$	3	65	yes	1	no	yes	12	375

# Non-monotonic Rule Systems - Example V

- $a_1$  is not acceptable because it has one bedroom only (rule  $r_2$ )
- $a_4$  and  $a_6$  because no pets are allowed (rule  $r_4$ )
- $a_2$  is more than the \$300 Carlos is willing to pay (rules  $r_7, r_9$ )
- $a_3, a_5, a_7$  are acceptable (rule  $r_1$ )

Carlos preferences are further based on price, garden size, and size, in that order.

$$r_{10} : \text{cheapest}(X) \implies \text{rent}(X)$$

$$r_{11} : \text{cheapest}(X), \text{largestGarden}(X) \implies \text{rent}(X)$$

$$r_{12} : \text{cheapest}(X), \text{largestGarden}(X), \text{largest}(X) \implies \text{rent}(X)$$

$$r_{12} > r_{10}$$

$$r_{12} > r_{11}$$

$$r_{11} > r_{10}$$

Also, we need to specify that at most one apartment can be rented, using conflict sets:

$$C(\text{rent}(X)) = \{\neg\text{rent}(X)\} \cup \{\text{rent}(Y) \mid Y \neq X\}$$

- *cheapest*( $a_3$ )
- *cheapest*( $a_5$ )
- *largest*( $a_3$ )
- *largest*( $a_7$ )
- *largestGarden*( $a_5$ )
  
- $r_{11}$  can be applied to  $a_5$
- $r_{10}$  can be applied to  $a_3$ , thus establishing an attack
- $r_{11}$  is stronger than  $r_{10}$ , attack is successfully countered
- no further attacks
- $a_5$  is the apartment of choice!

- 1 Rule Systems
  - Motivation
  - Rules
  - Facts and Logic Programs
  - Goals
- 2 Monotonic Rule Systems
  - Definition
  - Example
- 3 Non-monotonic Rule Systems
  - Motivation
  - Defeasible Rules
  - Example
- 4 Rule Markup
  - Rule Markup in XML: Monotonic Rules
  - Rule Markup in XML: Non-monotonic Rules
  - SWRL: A Semantic Web Rule Language Combining OWL and RuleML

# Rule Markup in XML: Monotonic Rules - Terms

Terms are represented using XML tags `<term>`, `<function>`, `<var>`, `<const>`.

For example,

$$f(X, a, g(b, Y))$$

is represented as

```
<term>
  <function> f </function>
  <term>
    <var> X </var>
  </term>
  <term>
    <const> a </const>
  </term>
  <term>
    <function> g </function>
    <term>
      <const> b </const>
    </term>
    <term>
      <var> Y </var>
    </term>
  </term>
</term>
```

# Rule Markup in XML: Monotonic Rules - Atomic Formulas

Additionally, the tags `<atom>` and `<predicate>` are used. E.g.  $f(X, a, g(b, Y))$ :

```
<atom>
  <predicate> f </predicate>
  <term>
    <var> X </var>
  </term>
  <term>
    <const> a </const>
  </term>
  <term>
    <predicate> g </predicate>
    <term>
      <const> b </const>
    </term>
    <term>
      <var> Y </var>
    </term>
  </term>
</atom>
```

A fact is just an atomic formula, enclosed in `<fact>`

```
<fact>
  <atom>
    <predicate> p </predicate>
    <term>
      <const> a </const>
    </term>
  </atom>
</fact>
```

# Rule Markup in XML: Monotonic Rules - Rules

A rule consists of a head and a body  $\implies$  `<rule>`, `<head>`,  
`<body>`.

$$p(X, a), q(Y, b) \longrightarrow r(X, Y)$$

```
<rule>
  <head>
    <atom>
      <predicate> r </predicate>
      <term>
        <var> X </var>
      </term>
      <term>
        <var> Y </var>
      </term>
    </atom>
  </head>
```

# Rule Markup in XML: Monotonic Rules - Rules II

```
<body>
  <atom>
    <predicate> p </predicate>
    <term>
      <var> X </var>
    </term>
    <term>
      <const> a </const>
    </term>
  </atom>
  <atom>
    <predicate> q </predicate>
    <term>
      <var> Y </var>
    </term>
    <term>
      <const> b </const>
    </term>
  </atom>
</body>
</rule>
```

## Queries

Queries are represented as bodies of rules, surrounded by <query> tags.

## DTD for monotonic rules

A program consists of a number of rules and facts.

```
<!ELEMENT program ( (rule|fact)* ) >
```

A fact consists of an atomic formula

```
<!ELEMENT fact (atom) >
```

A rule consists of a head and a body

```
<!ELEMENT rule (head,body) >
```

A head consists of an atomic formula

```
<!ELEMENT head (atom) >
```

A body is a list of atomic formulas

```
<!ELEMENT body (atom*) >
```

An atomic formula consists of a predicate, followed by a number of terms

```
<!ELEMENT atom (predicate,term*) >
```

A term is a constant, a variable, or a composite term consisting of a function symbol, followed by a number of terms

```
<!ELEMENT term (const|var|(function,term*)) >
```

Predicates, function symbols, constants, and variables are atomic types:

```
<!ELEMENT predicate (#PCDATA)>
```

```
<!ELEMENT function (#PCDATA)>
```

```
<!ELEMENT const (#PCDATA)>
```

```
<!ELEMENT var (#PCDATA)>
```

A query is a list of atomic formulas

```
<!ELEMENT query (atom*) >
```

# Rule Markup in XML: Non-monotonic Rules

Compared to monotonic rules, non-monotonic rules have the following syntactic differences:

- Negated atoms may occur in the head and the body of a rule
- each rule has a label
- apart from rules and facts, a program also contains priority statements

## Example (Non-monotonic rules in XML)

*Consider the defeasible program*

$$\begin{array}{l} r_1 : p(X) \implies s(X) \\ r_2 : q(X) \implies \neg s(X) \\ p(a) \\ q(a) \\ r_1 > r_2 \end{array}$$

# Rule Markup in XML: Non-monotonic Rules - Rules

Rule  $r_1$  is represented as follows:

```
<rule id="r1">
  <head>
    <atom>
      <predicate> s </predicate>
      <term>
        <var> X </var>
      </term>
    </atom>
  </head>
  <body>
    <atom>
      <predicate> p </predicate>
      <term>
        <var> X </var>
      </term>
    </atom>
  </body>
</rule>
```

Rule  $r_2$  is represented accordingly.

The fact  $p(a)$  is represented as follows:

```
<fact>
  <atom>
    <predicate> p </predicate>
    <term>
      <const> a </const>
    </term>
  </atom>
</fact>
```

The priority relation  $r_1 > r_2$  is represented as follows:

```
<stronger superior="r1" inferior="r2">
```

## DTD

A program consists of a number of rules, facts, and priority relations.

```
<!ELEMENT program ( (rule|fact|stronger)* ) >
```

A fact consists of an atomic formula or its negation

```
<!ELEMENT fact (atom|neg) >
```

```
<!ELEMENT neg (atom) >
```

A rule consists of a head and a body, and an id attribute

```
<!ELEMENT rule (head,body) >
```

```
<!ATTLIST rule  
    id ID #IMPLIED >
```

A rule head and body are defined as for monotonic rules, but may contain negated atoms.

```
<!ELEMENT head (atom|neg) >
```

```
<!ELEMENT body ((atom|neg)*) >
```

An atomic formula consists of a predicate, followed by a number of variables or constants

```
<!ELEMENT atom (predicate,(var|const)*) >
```

A priority element uses two attributes, referring to the superior and the inferior rule

```
<!ELEMENT stronger EMPTY>
```

```
<!ATTLIST stronger
```

```
    superior IDREF #REQUIRED >
```

```
    inferior IDREF #REQUIRED >
```

Predicates, constants, and variables are atomic types:

```
<!ELEMENT predicate (#PCDATA)>
```

```
<!ELEMENT const (#PCDATA)>
```

```
<!ELEMENT var (#PCDATA)>
```

A query is a list of atomic formulas

```
<!ELEMENT query (atom*) >
```

- SWRL: Semantic Web Rule Language
- SWRL is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language.
- extends the set of OWL axioms to include Horn-like rules
- It thus enables Horn-like rules to be combined with an OWL knowledge base

## XML - concrete Syntax:

```
<swrlx:Ontology swrlx:name = xsd:anyURI>
```

```
Content: (
```

```
owlx:VersionInfo           | owlx:PriorVersion |
owlx:Annotation            | owlx:IncompatibleWith |
owlx:Imports                | owlx:BackwardCompatibleWith |
owlx:Class[axiom]          | owlx:EnumeratedClass(D,F) |
owlx:SubClassOf(D,F)       | owlx:EquivalentClasses |
owlx:DisjointClasses(D,F)  | owlx:DatatypeProperty |
owlx:ObjectProperty        | owlx:SubPropertyOf |
owlx:EquivalentProperties  | owlx:Individual[axiom] |
owlx:SameIndividual        | owlx:DifferentIndividuals |
ruleml:imp[axiom]          | ruleml:var[axiom])*
```

```
</swrlx:Ontology>
```

*Remark: owlx: XML-Syntax von OWL*

## Example (SWRL-Examples in XML concrete syntax)

*Assert that the combination of the hasParent and hasBrother properties implies the hasUncle property in SWRL:*

```
<ruleml:imp> <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

## Example

*Alternatively, the hasUncle property can be assert if x1 hasParent x2, x2 hasSibling x3, and x3 hasSex male, then x1 hasUncle x3:*

```
<ruleml:imp>
  <ruleml:_rlabel ruleml:href="#example2"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasSibling">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasSex">
      <ruleml:var>x3</ruleml:var>
      <owlx:Individual owlx:name="#male" />
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
```

## Example

```
<ruleml:_head>
  <swrlx:individualPropertyAtom
    swrlx:property="hasUncle">
    <ruleml:var>x1</ruleml:var>
    <ruleml:var>x3</ruleml:var>
  </swrlx:individualPropertyAtom>
</ruleml:_head>
</ruleml:imp>
```

## Example (SWRL-Examples in RDF Concrete Syntax)

```
<swrl:Variable rdf:ID="x1"/>
<swrl:Variable rdf:ID="x2"/>
<swrl:Variable rdf:ID="x3"/>
<ruleml:imp>
  <ruleml:body rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasParent"/>
      <swrl:argument1 rdf:resource="#x1" />
      <swrl:argument2 rdf:resource="#x2" />
    </swrl:IndividualPropertyAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasSibling"/>
      <swrl:argument1 rdf:resource="#x2" />
      <swrl:argument2 rdf:resource="#x3" />
    </swrl:IndividualPropertyAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasSex"/>
      <swrl:argument1 rdf:resource="#x3" />
      <swrl:argument2 rdf:resource="#male" />
    </swrl:IndividualPropertyAtom>
  </ruleml:body>
```

## Example (SWRL-Examples in RDF Concrete Syntax)

```
<ruleml:head rdf:parseType="Collection">
  <swrl:IndividualPropertyAtom>
    <swrl:propertyPredicate rdf:resource="&eg;hasUncle"/>
    <swrl:argument1 rdf:resource="#x1" />
    <swrl:argument2 rdf:resource="#x3" />
  </swrl:IndividualPropertyAtom>
</ruleml:head>
</ruleml:imp>
```